IFDS Framework

CMPUT 497/500
Foundations of Program Analysis

Karim Ali
@karimhamdanali

Previously

- Inter-Procedural Data-Flow

- Inherited vs Synthesized Analysis Info

- Caller-Callee Relationships

- Valid/Invalid Paths

- Staircase of Calls and Returns

- Demand-Driven Analysis

- Types of Contexts

- Important Language Features

Interprocedural Finite Distributive Subset Problems

Thomas Reps, Susan Horwitz, and Mooly Sagiv. Precise Interprocedural Dataflow Analysis via Graph Reachability. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '95), pages 49–61.
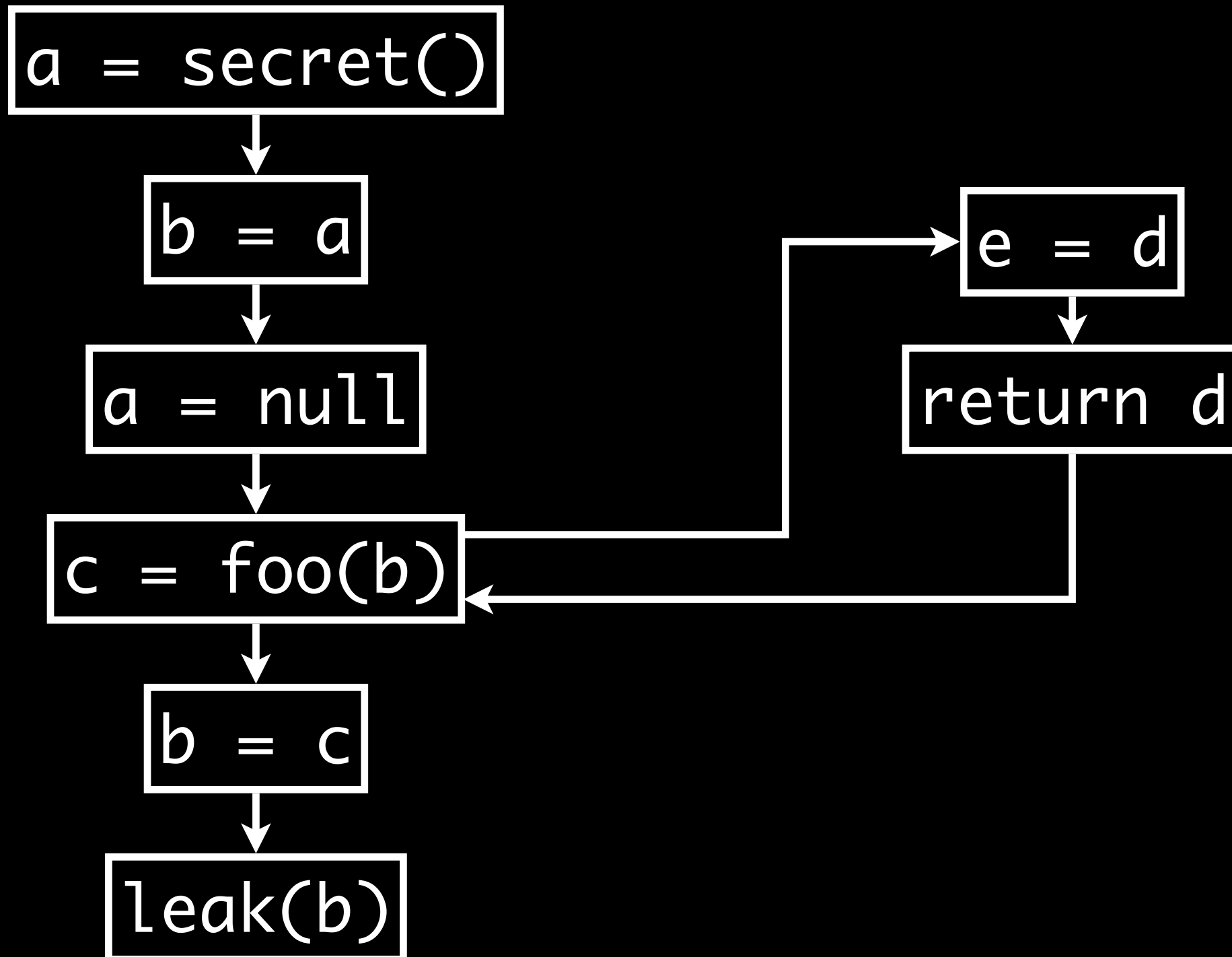
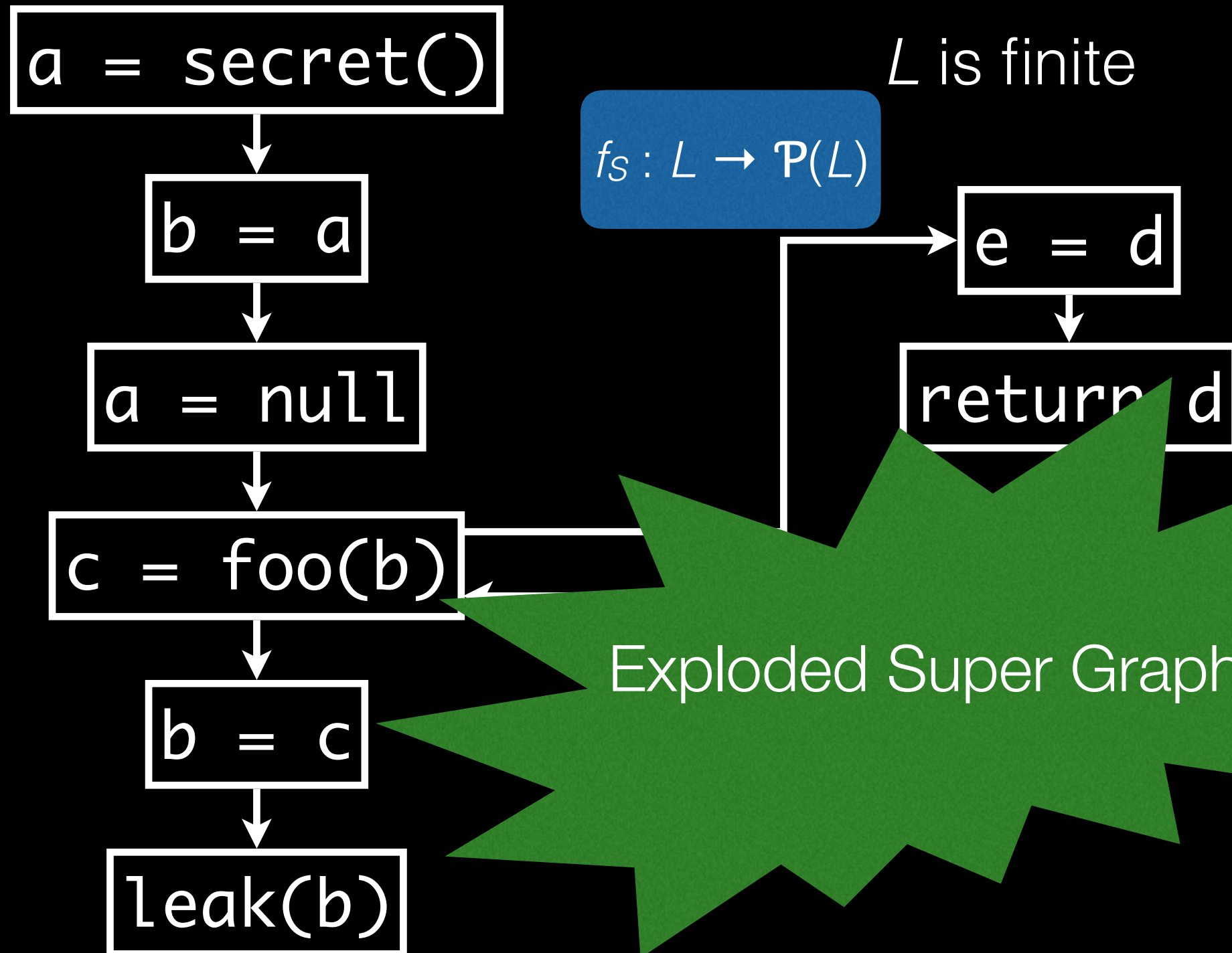**I**nterprocedural **F**inite **D**istributive **S**ubset Problems

Interprocedural

```
main() {
  a = secret();
  b = a;
  a = null;
  c = foo(b);
  b = c;
  leak(b);
}
```

```
foo(d) {
  e = d;
  return d;
}
```
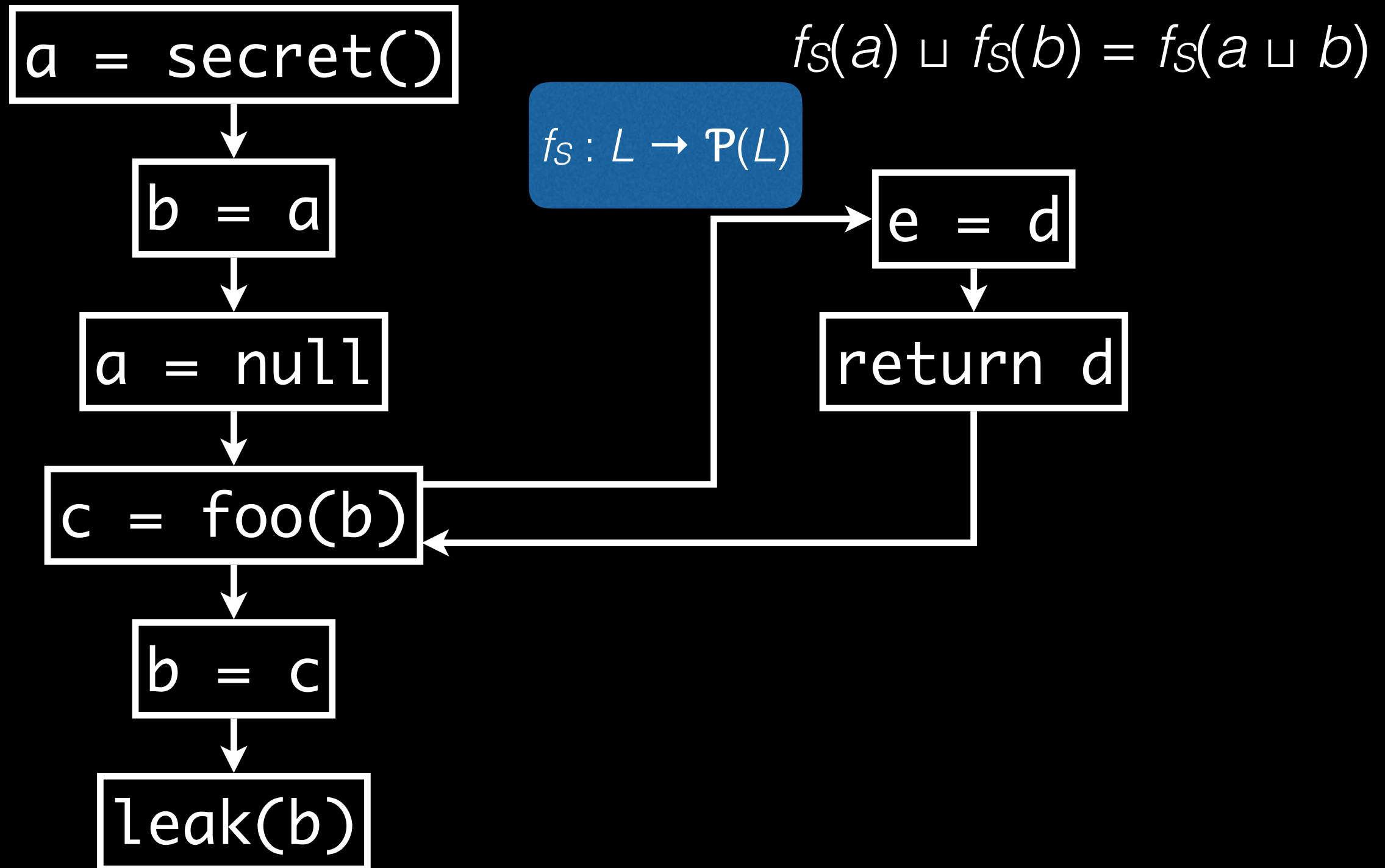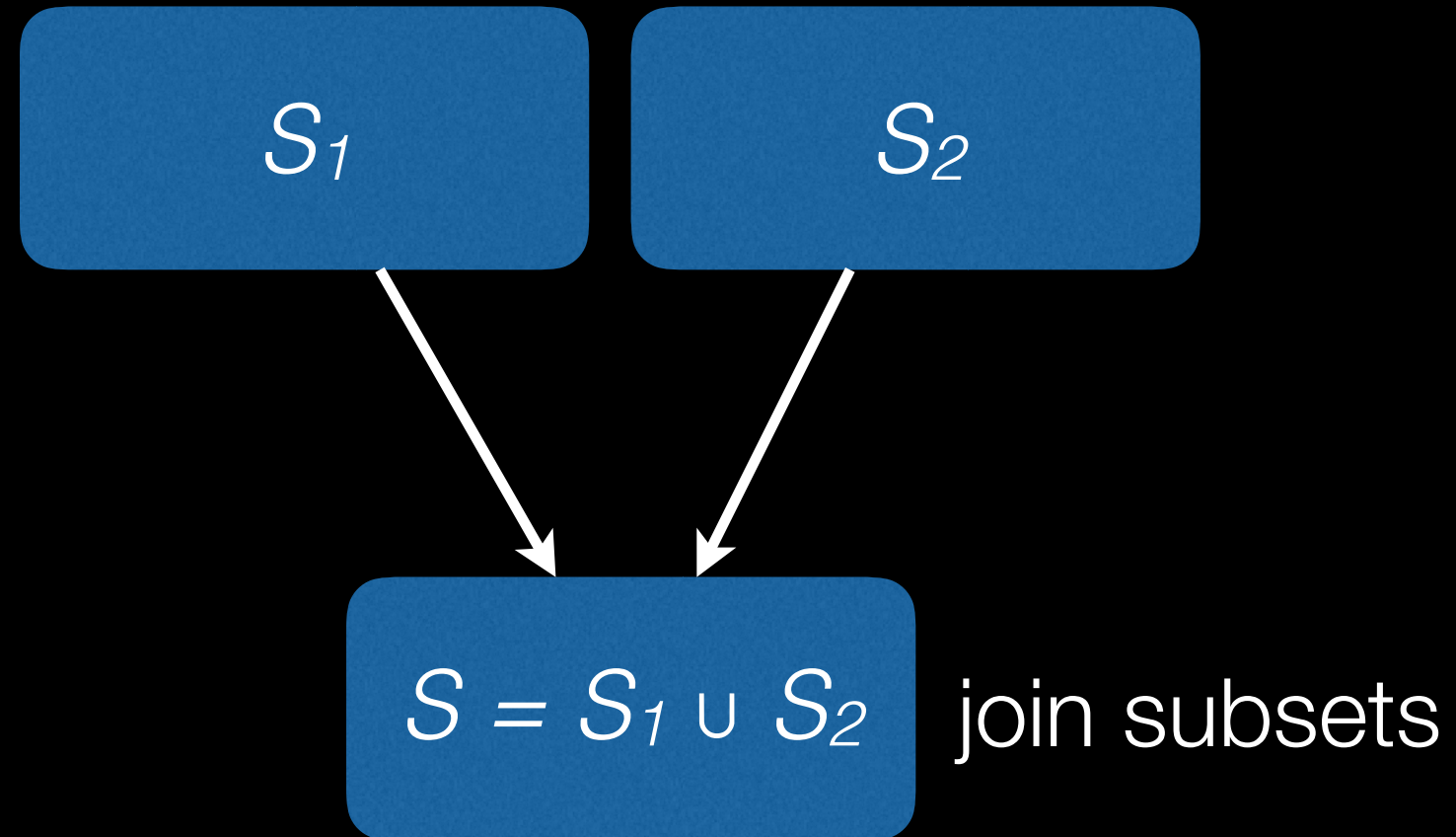
Interprocedural


```
a = secret()
```

```
b = a
```

```
a = null
```

```
c = foo(b)
```

```
b = c
```

```
leak(b)
```

```
e = d
```

```
return d
```

Finite

```
a = secret()
```

```
b = a
```

L is finite

$f_S : L \rightarrow \mathcal{P}(L)$

```
a = null
```

```
e = d
```

```
return d
```

```
c = foo(b)
```

Exploded Super Graph

```
b = c
```

```
leak(b)
```

Distributive

$$f_S(a) \sqcup f_S(b) = f_S(a \sqcup b)$$

$$f_S : L \to \mathcal{P}(L)$$

```
a = secret()
b = a
a = null
c = foo(b)
b = c
leak(b)
e = d
return d
```

Subset

S_1

S_2

$S = S_1 \cup S_2$ join subsets

alias　　　　　taint　　　　　typestate

IFDS

constant　　　reaching　　　uninitialized
propagation　　definitions　　variables

Graph Reachability

Graph Reachability

`0 a b c`

`a = secret()`

`b = a`

`a = null`

`c = foo(b)`

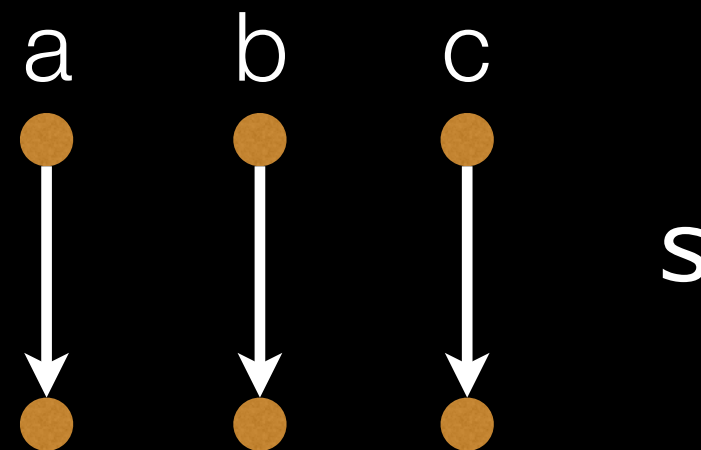`b = c`

`leak(b)`
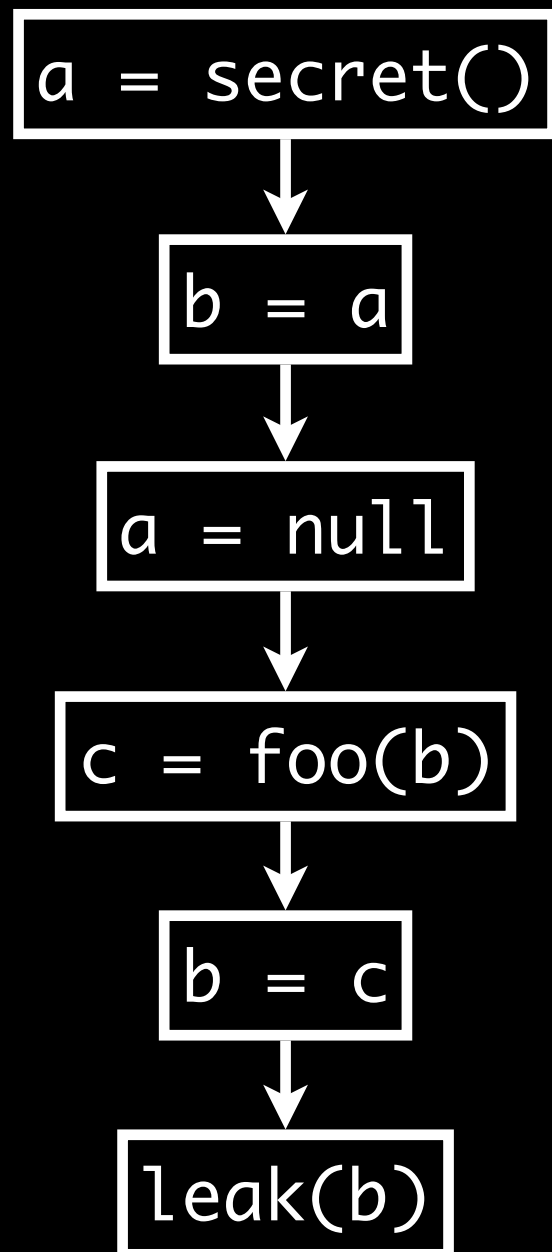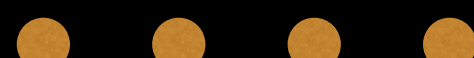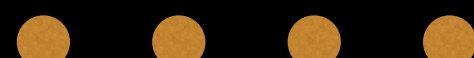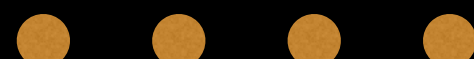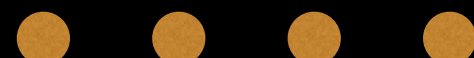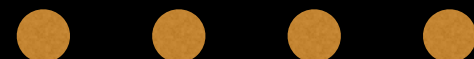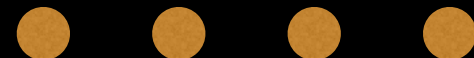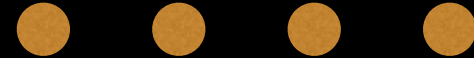
`0 d e`

`e = d`

`return d`

- each node is a fact
- fact holds at a statement => node is reachable in ESG

Examples of Flow Functions

Identity Flow Functions

in(s) = facts before s

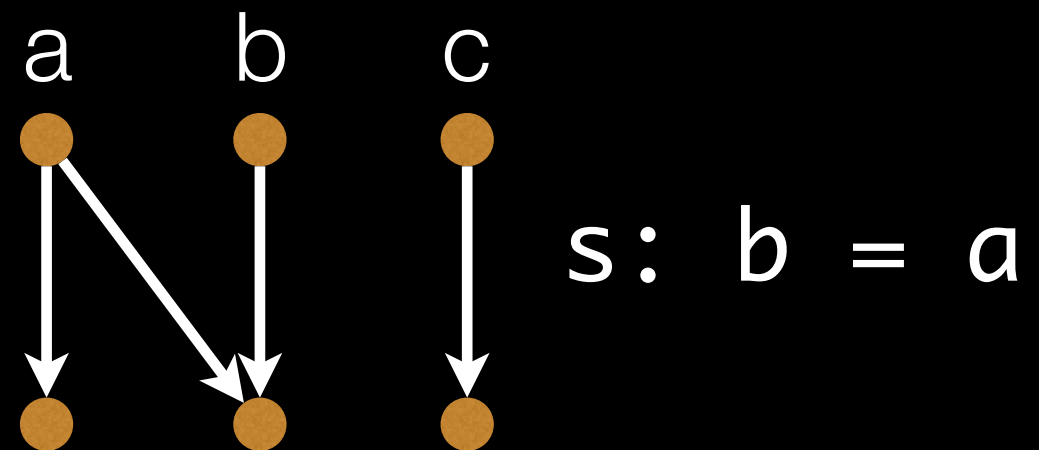a   b   c

s

out(s) = facts after s

every fact is reachable iff it was previously reachable

Flow Functions

$$\text{out}(\mathbf{s}) = \begin{cases} \text{in}(\mathbf{s}) \cup \{b\} \text{ if } f \in \text{in}(\mathbf{s}) \\ \\ \text{in}(\mathbf{s}), \text{ otherwise} \end{cases}$$
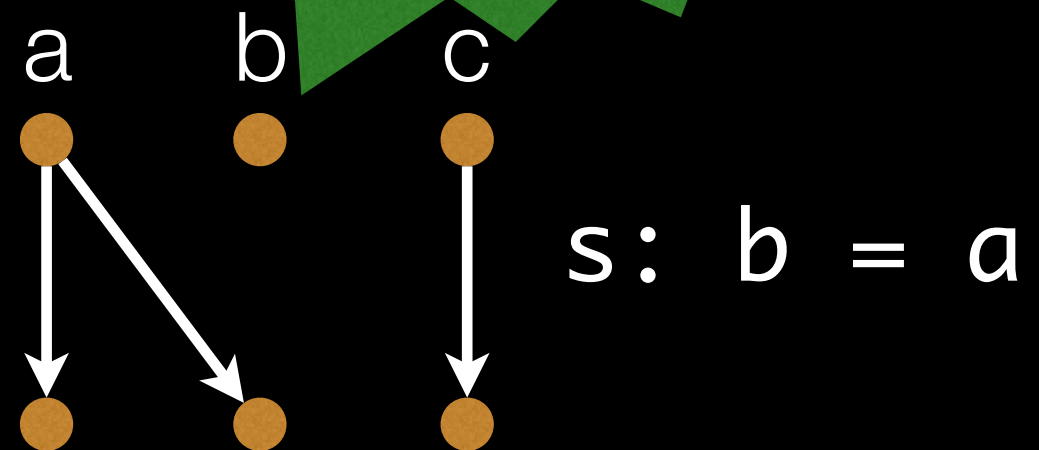
a   b   c

$s: \; b \; = \; a$

every fact is reachable iff it was previously reachable, and
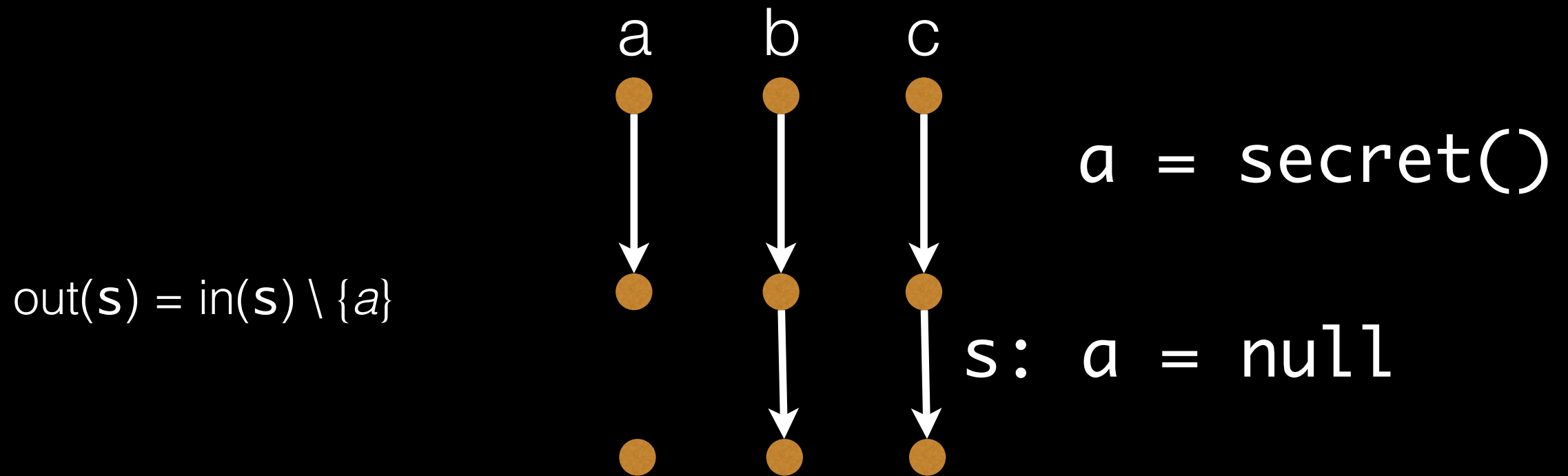*b* is also reachable if *a* was reachable

Flow Functions

Taint Analysis

$$out(s) = \begin{cases} in(s) \cup \{b\} \text{ if } f \in in(s) \\ in(s) \setminus \{b\}, \textit{otherwise} \end{cases}$$

a   b   c

$s: \ b \ = \ a$

every fact except *b* is reachable iff it was previously reachable; *b* is reachable if *a* was reachable
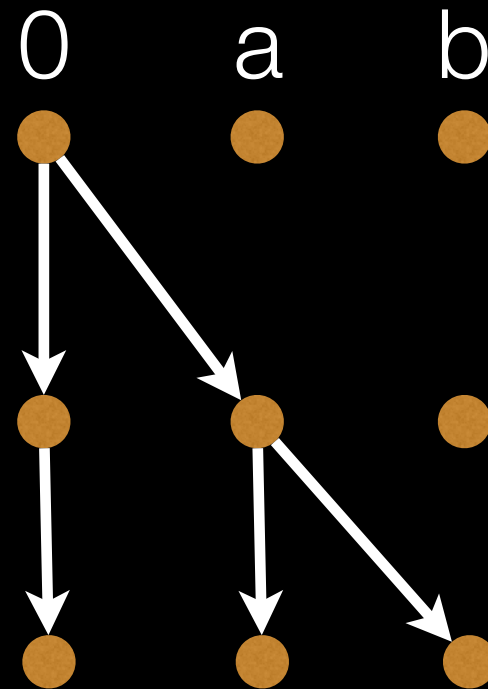
"Killing" Facts

a b c

$a = secret()$

$out(s) = in(s) \setminus \{a\}$

$s: a = null$

a is not reachable, even if it was before

Generating Facts

0   a   b

$\mathrm{out}(s) = \mathrm{in}(s) \cup \{a\}$

$a = \texttt{secret()}$

$\mathrm{out}(s) = \mathrm{in}(s) \cup \{b\}$

$b = a$

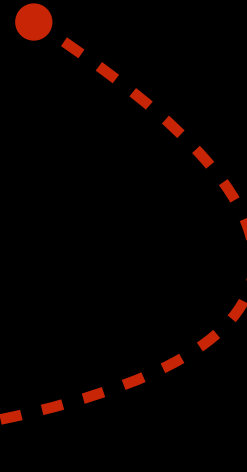*0* is the tautological fact => always reachable

Non-Distributive Flow Functions

a   b   c

¯_(ツ)_/¯
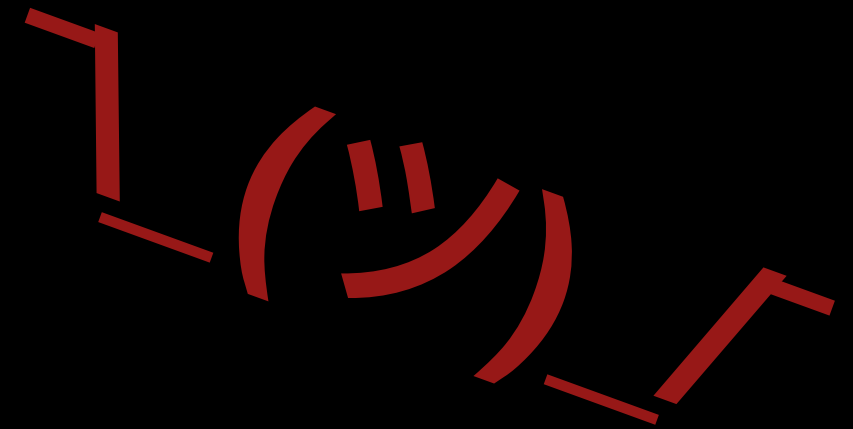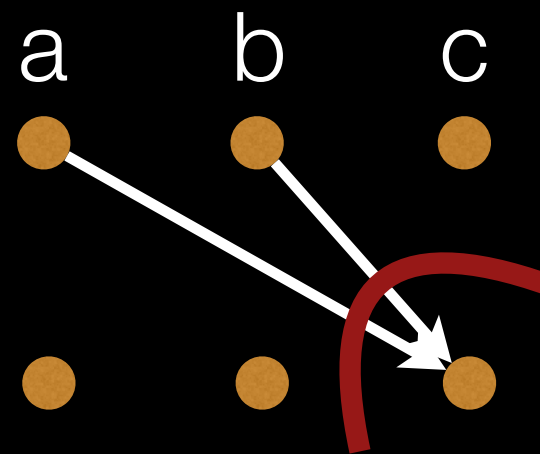
e.g., full constant propagation
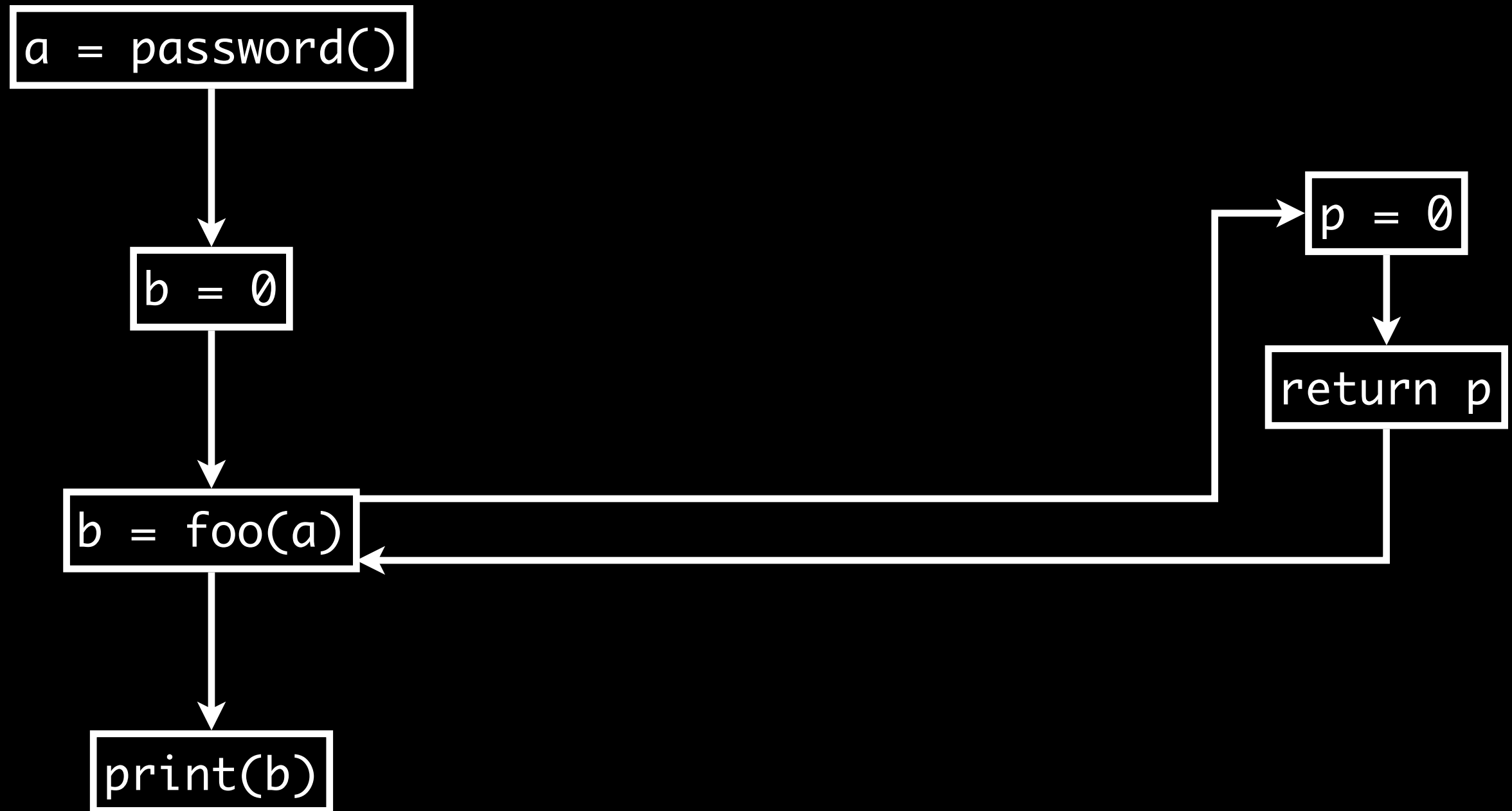c = a + b

Taint Analysis Example

```
void main() {
    int a = password();
    int b = 0;

    b = foo(a);

    print(b);
}
```
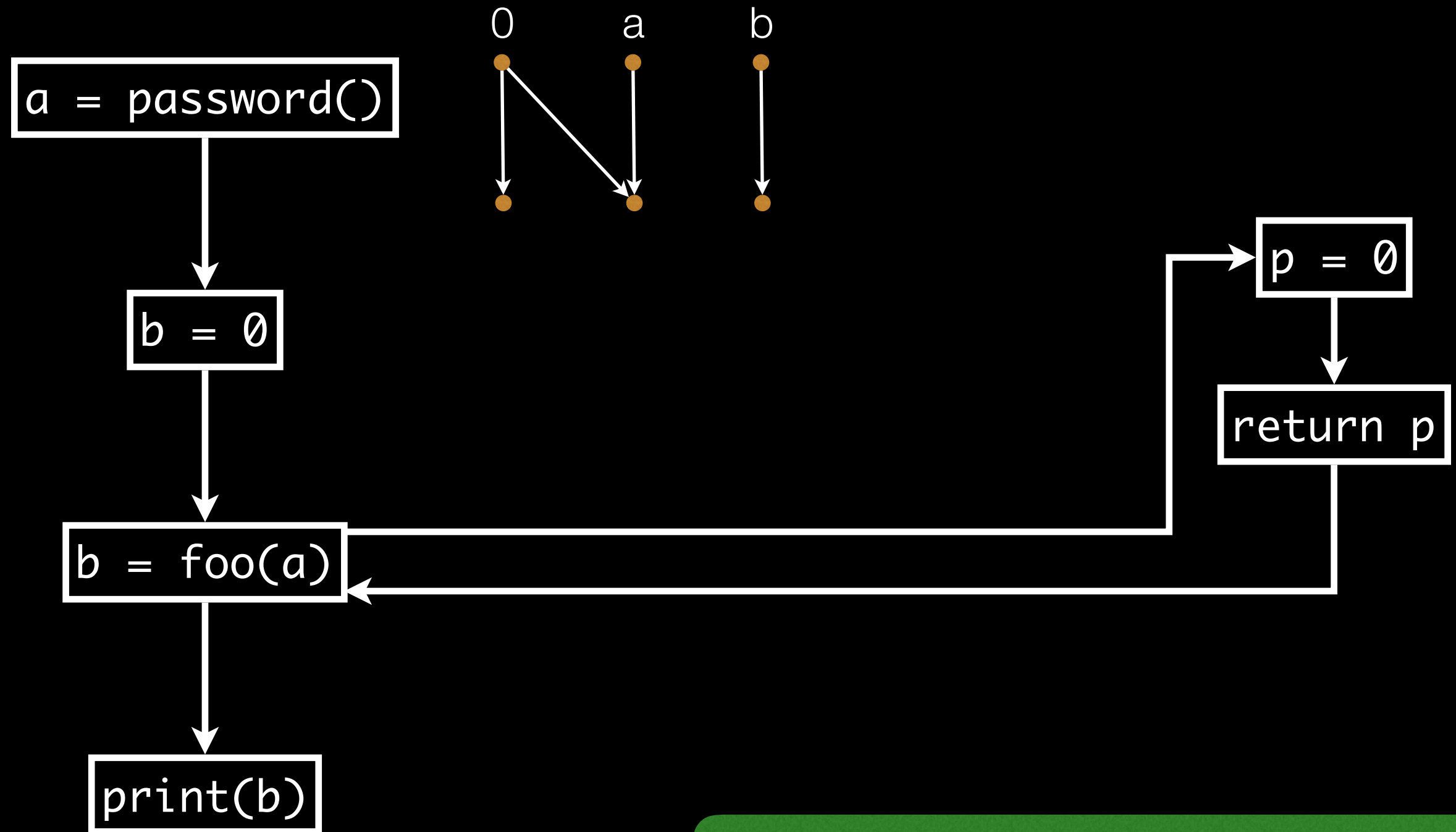
```
int foo(int p) {
    p = 0;

    return p;
}
```

Taint Analysis Example


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

Taint Analysis Example

0   a   b

```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

Normal-Flow Function

Taint Analysis Example

0   a   b

```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

still Normal-Flow Function

Taint Analysis Example

0   a   b

0   p

```
a = password()
```

```
b = 0
```

```
p = 0
```

```
return p
```

```
b = foo(a)
```

```
print(b)
```

Call-Flow Function

@karimhamdanali   24

Taint Analysis Example


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a   b

0   p

Normal-Flow Function

Taint Analysis Example


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a   b

0   p

Return-Flow Function

Taint Analysis Example

0   a   b

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

0   p

`p = 0`

`return p`

Normal-Flow Function

Taint Analysis Example

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`
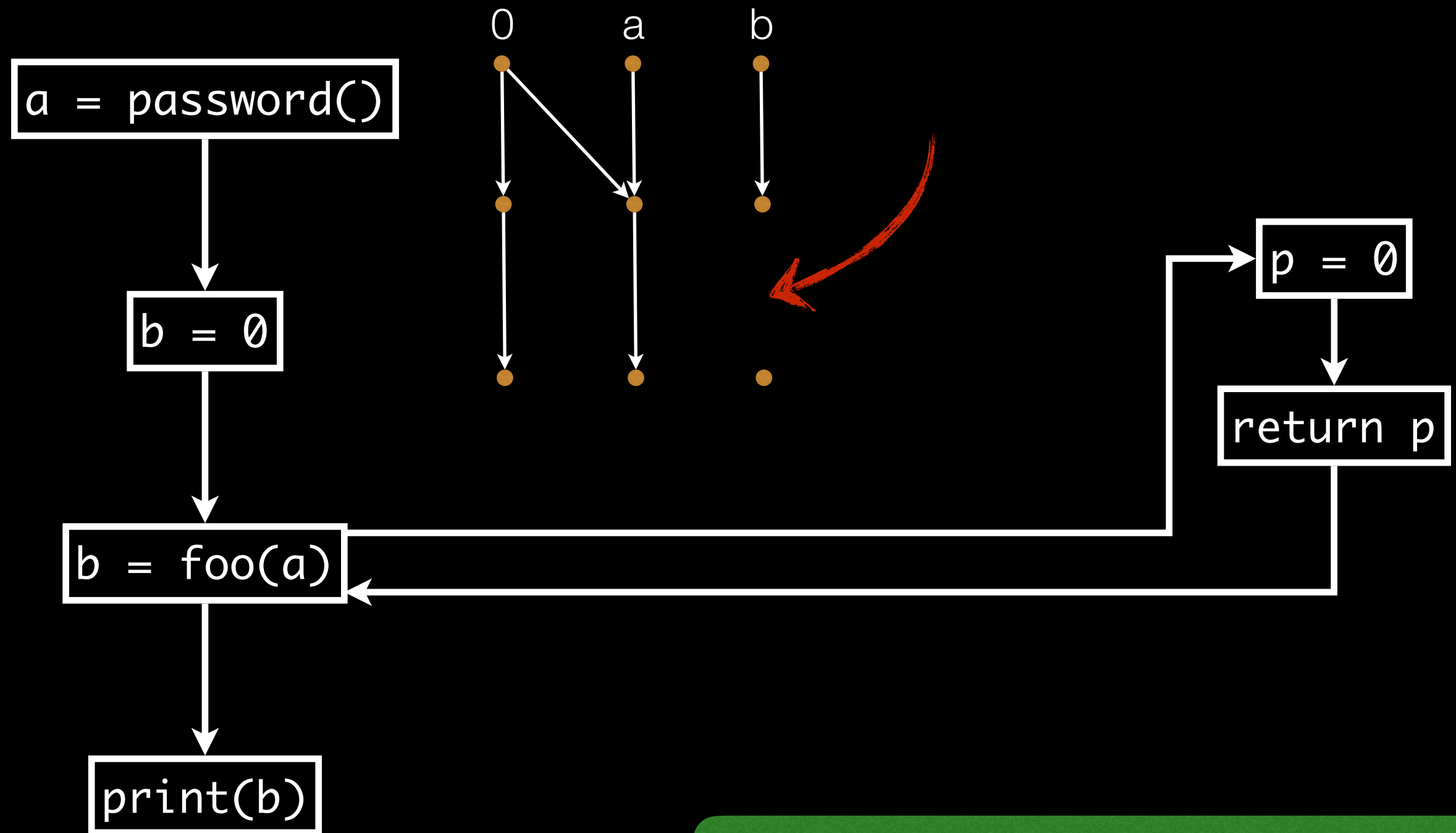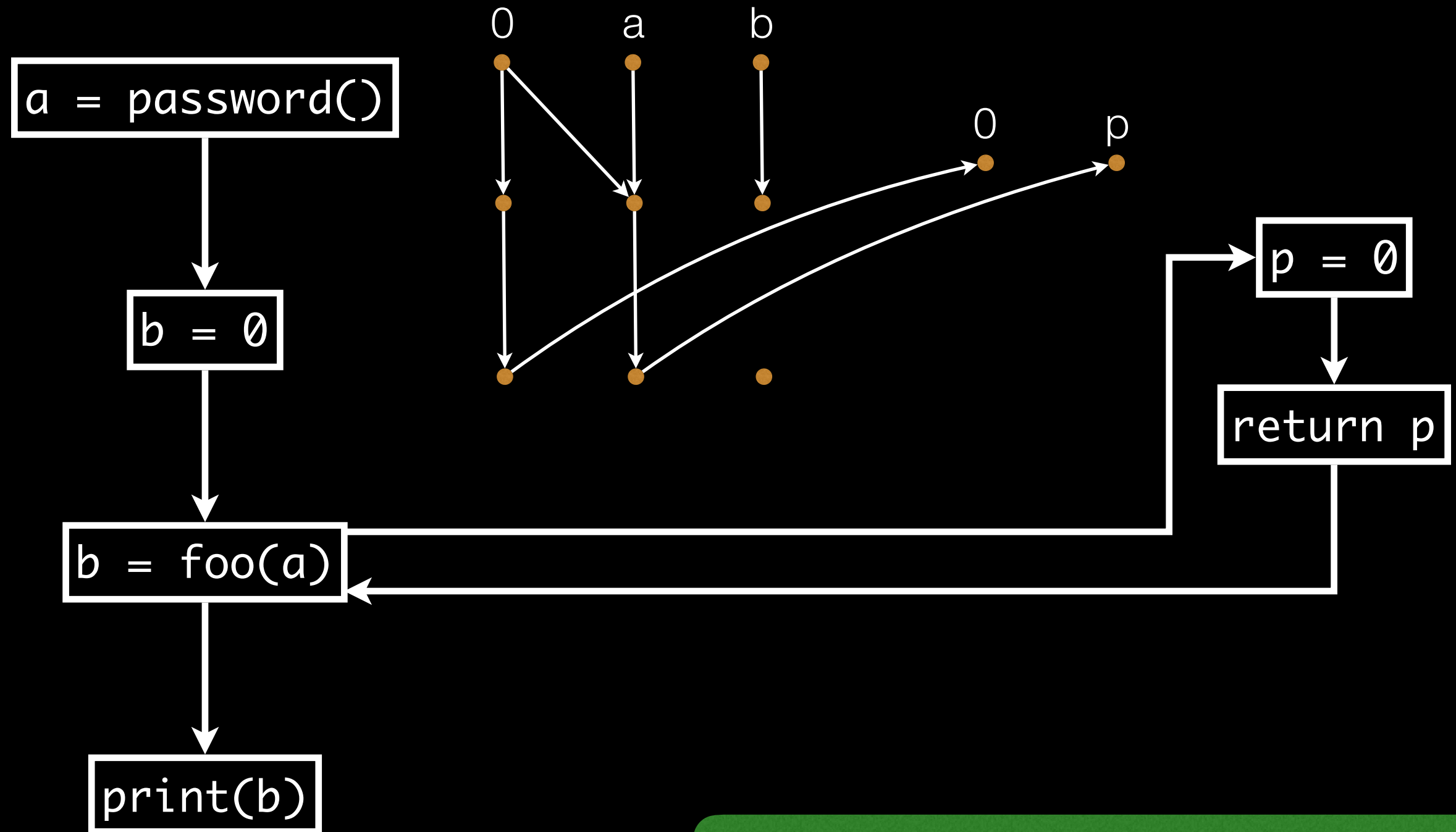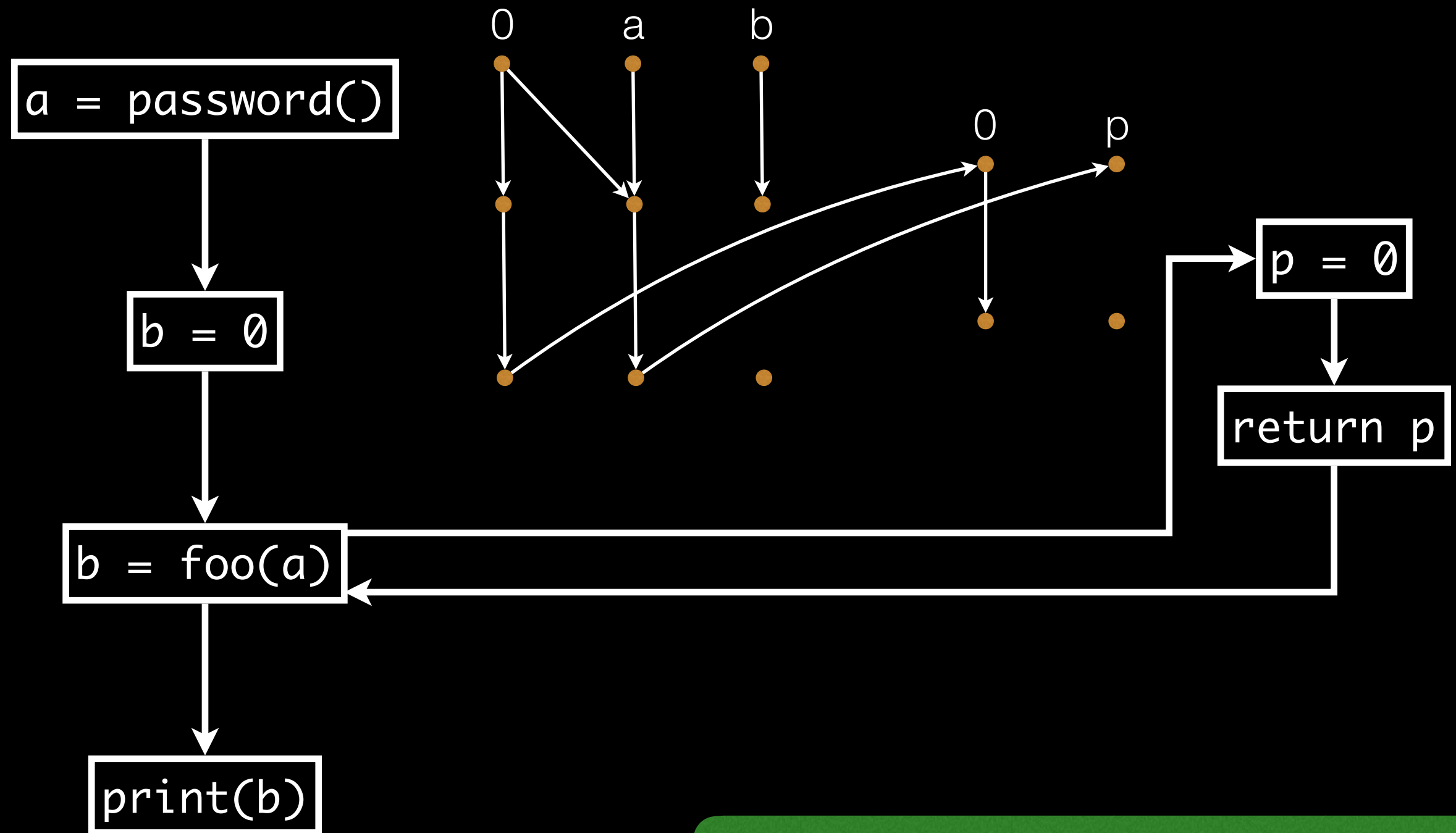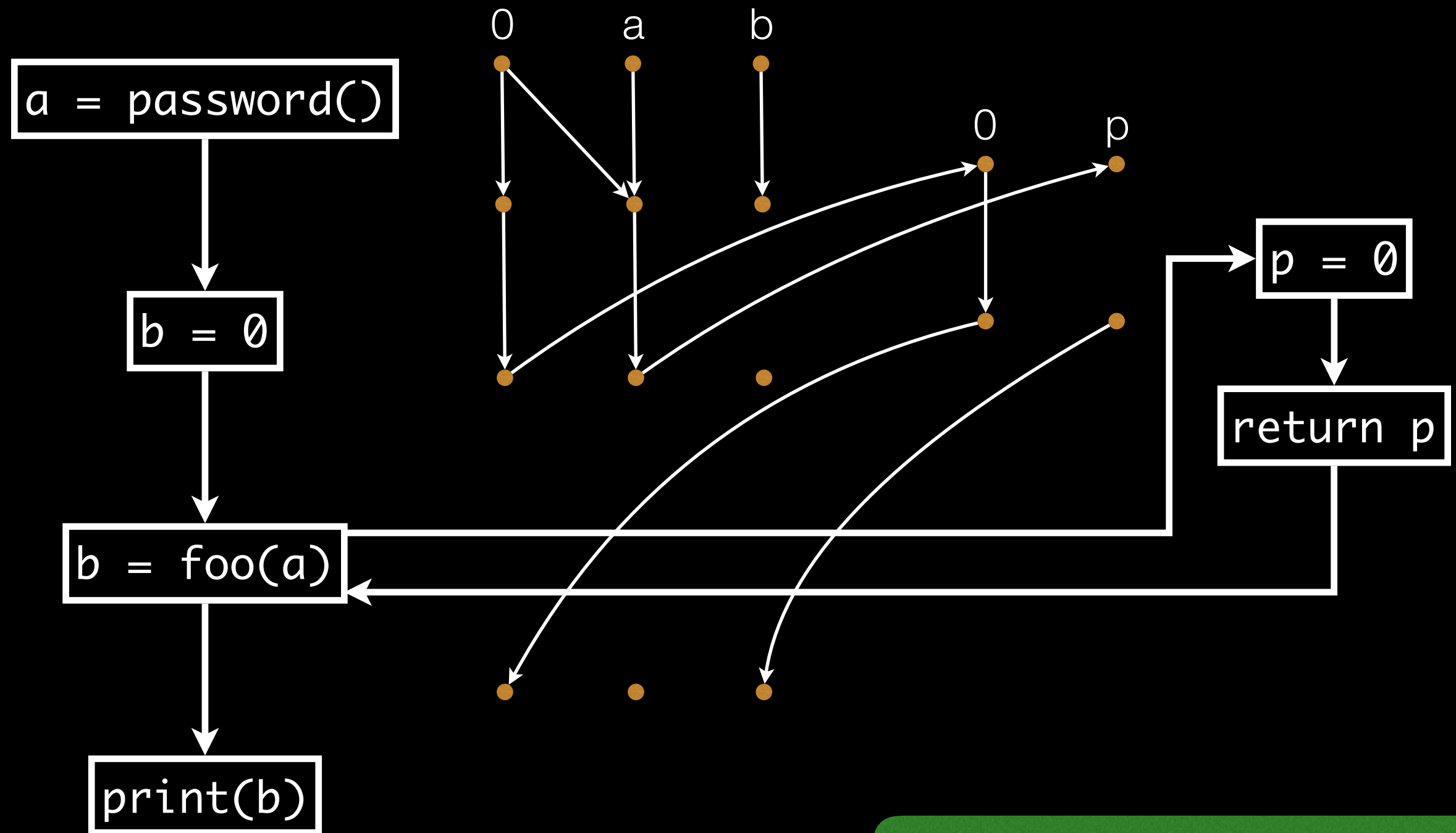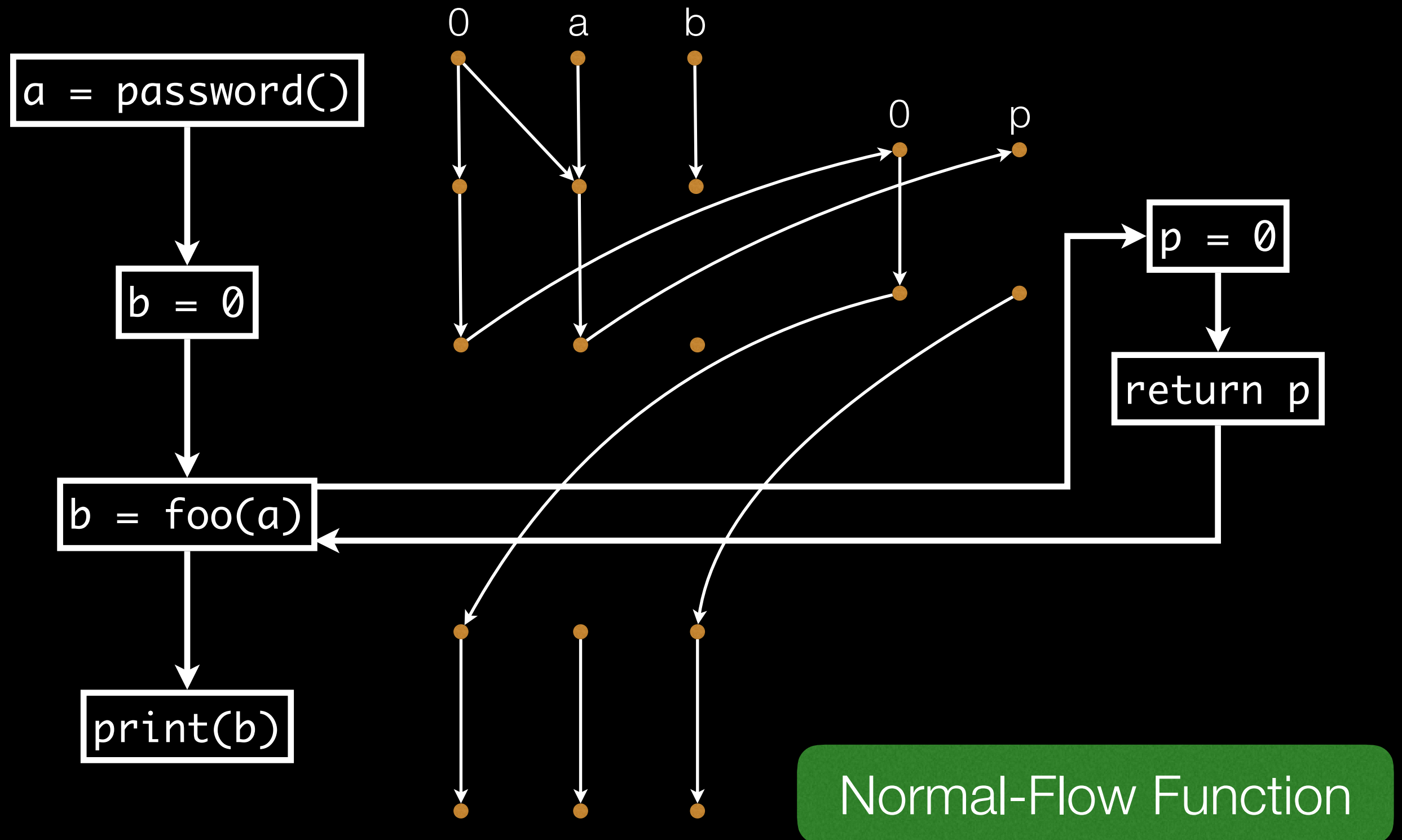
`return p`

Call-to-return-Flow Function
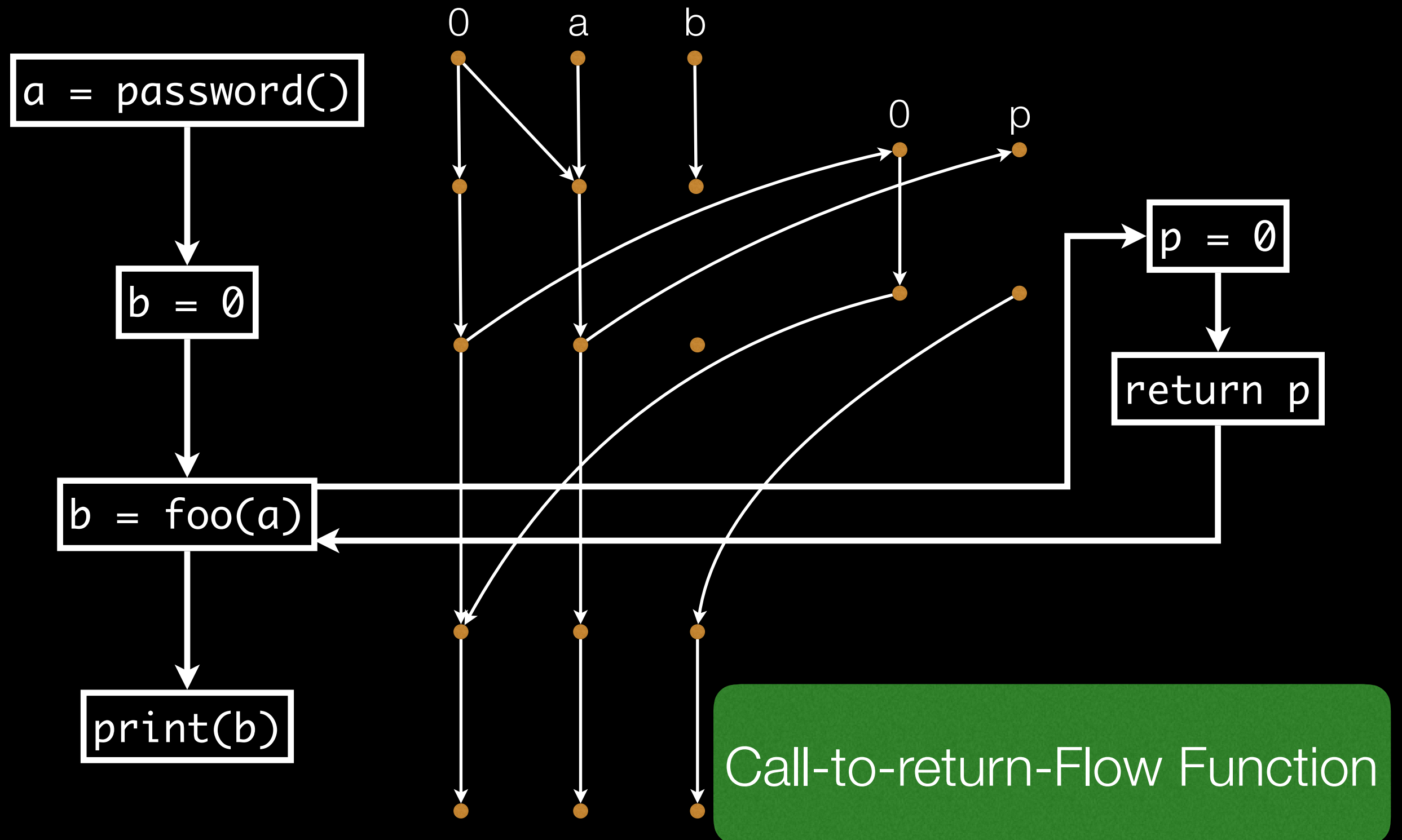
Flow Functions

Call-Flow Function

Normal-Flow Function

Return-Flow Function

id

Call-to-return-Flow Function

On-the-fly ESG

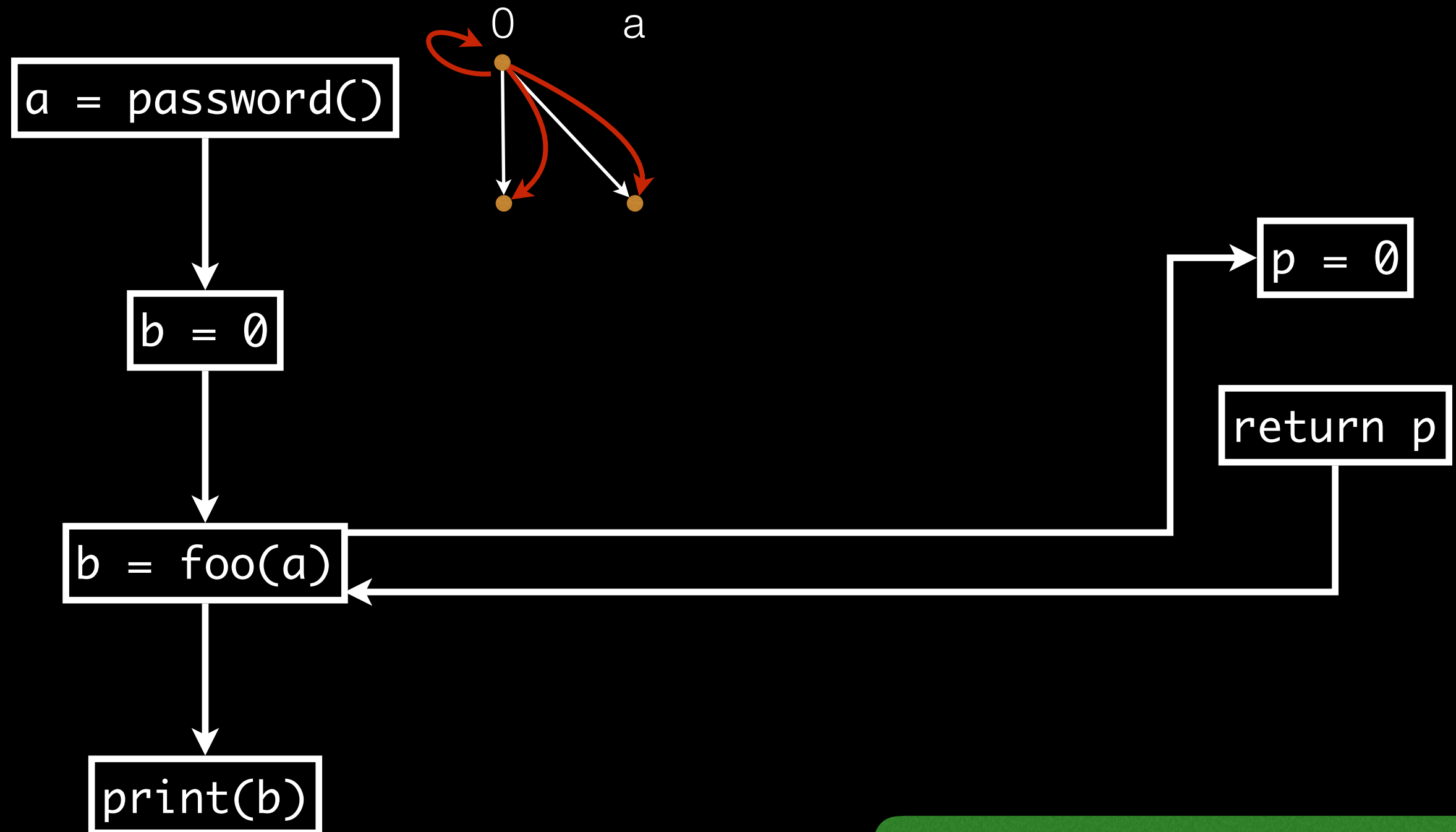# On-the-fly ESG

0

```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```
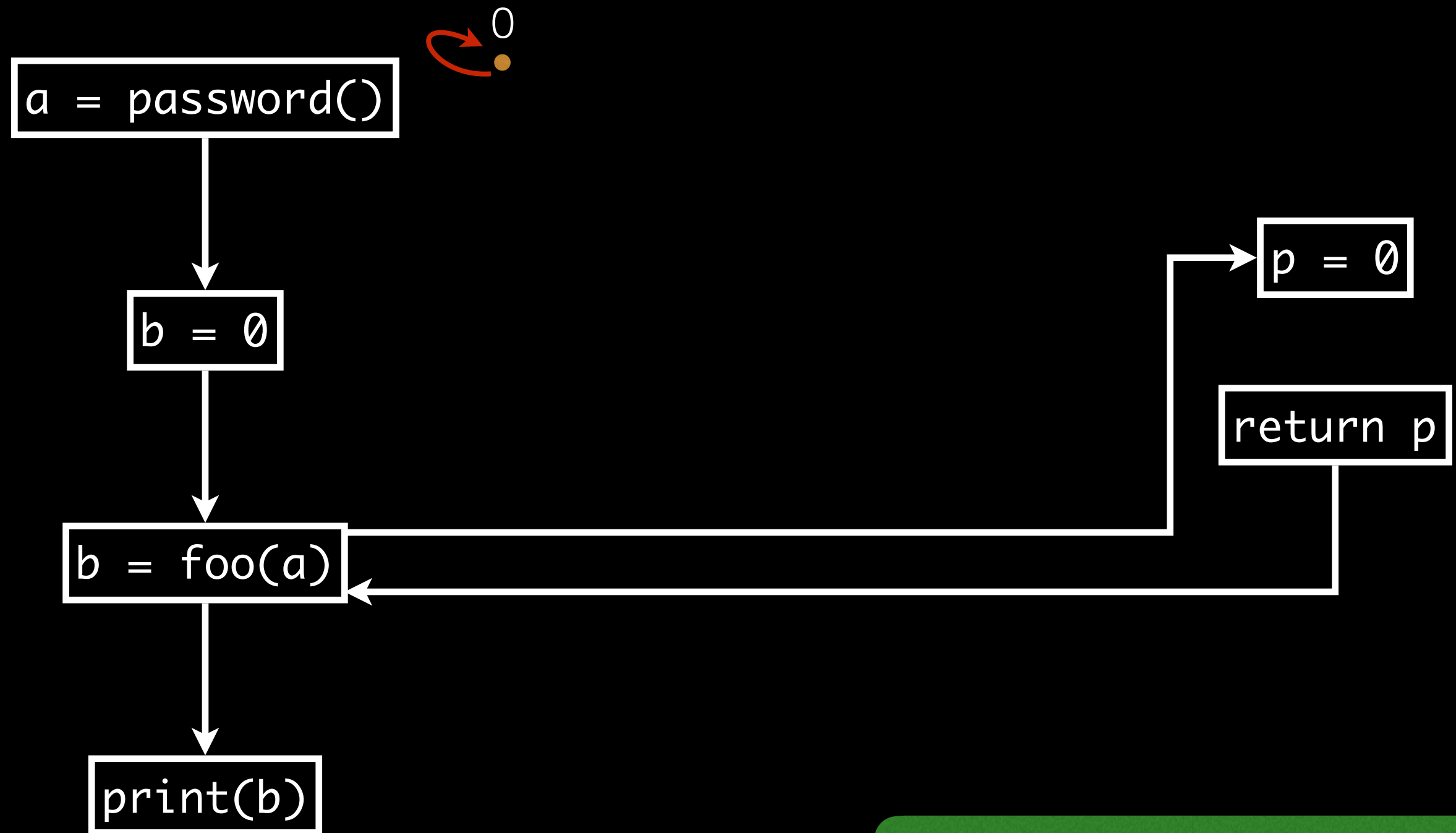
initial self-loop edge

On-the-fly ESG
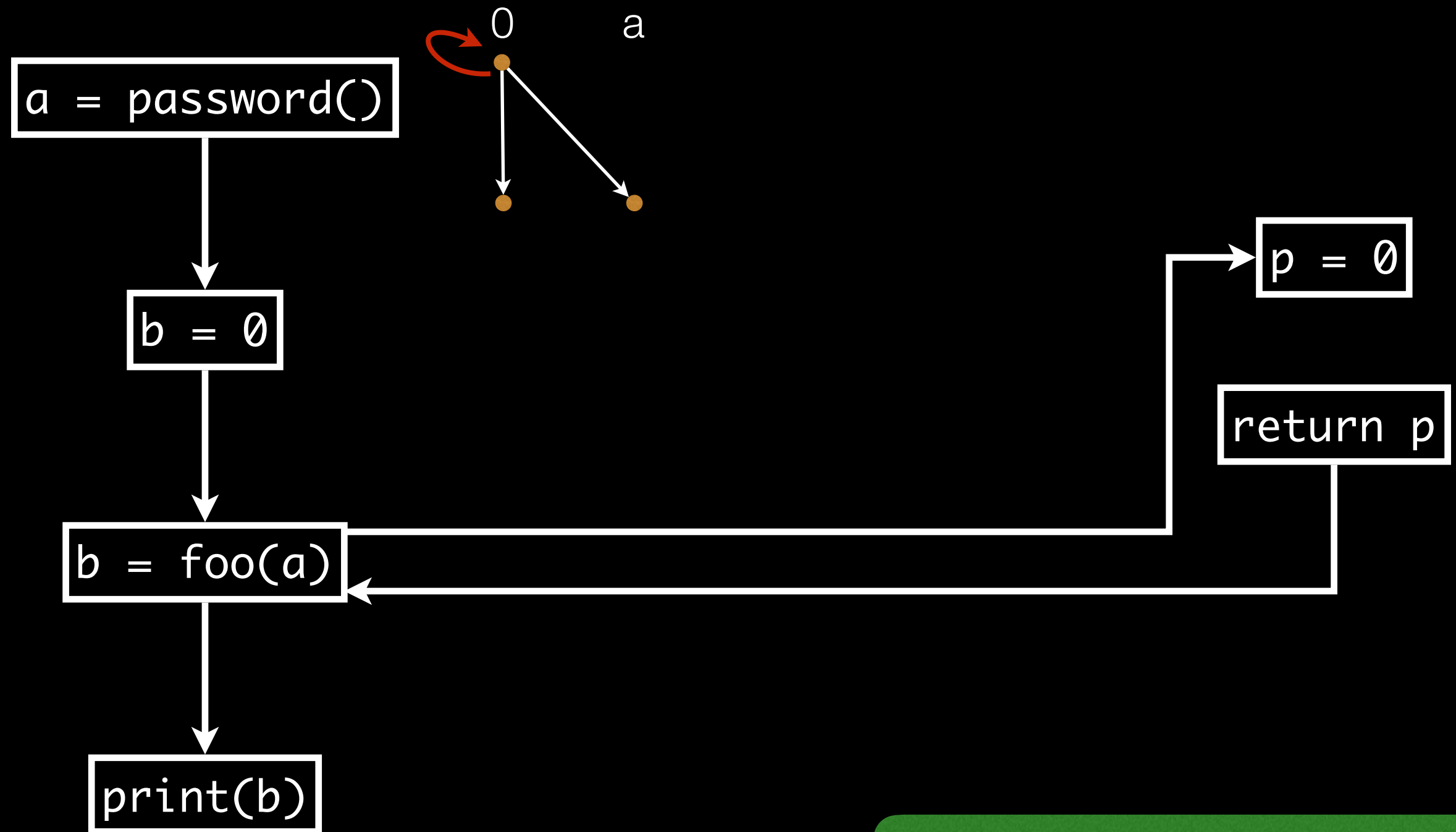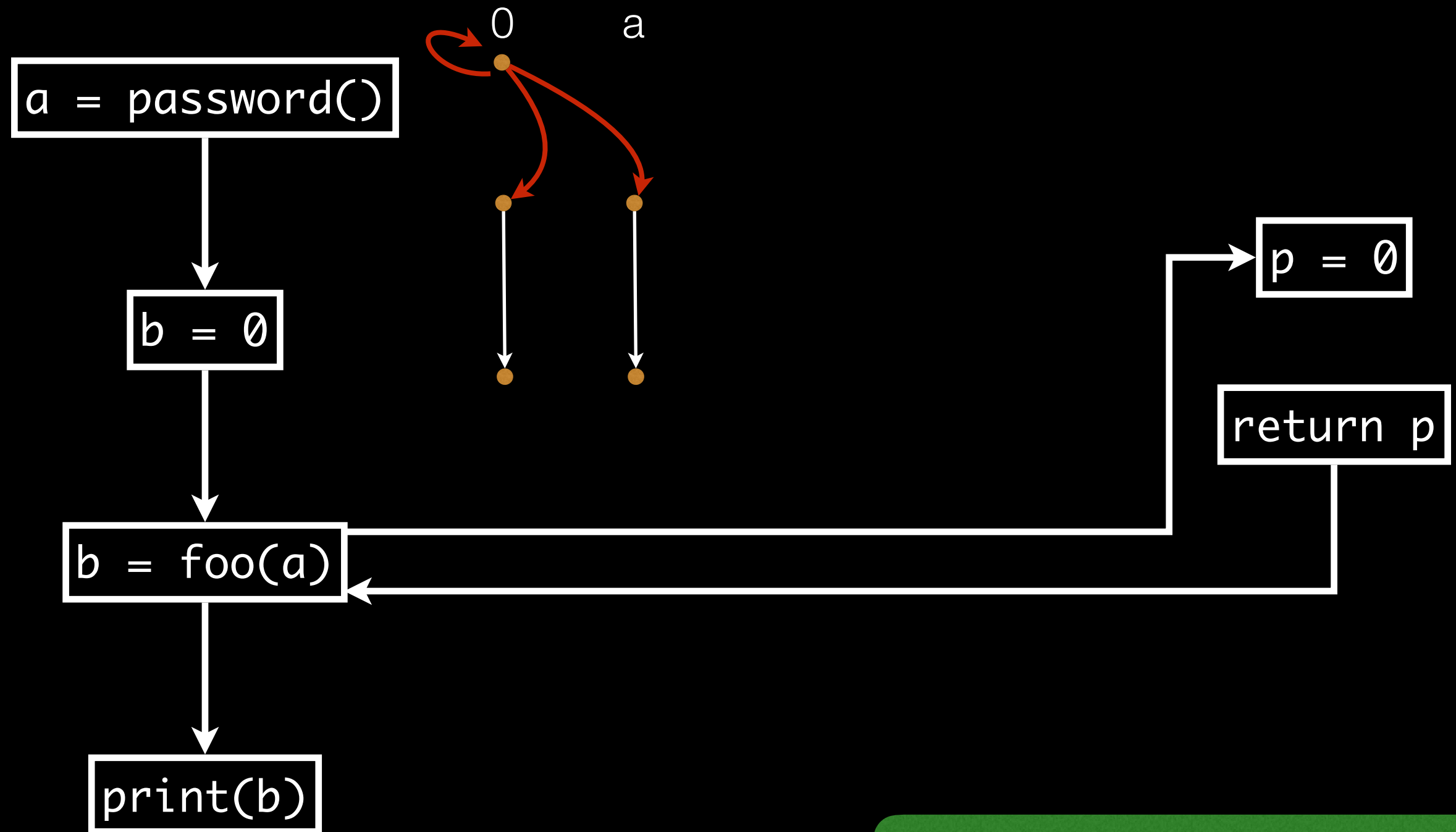
0   a

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`

`return p`

Normal-Flow

On-the-fly ESG

0   a

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`

`return p`

Path Edges

On-the-fly ESG


```
a = password()
```

0   a

```
b = 0
```

```
p = 0
```

```
b = foo(a)
```

```
return p
```

```
print(b)
```

Normal-Flow

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

0   a

`p = 0`

`return p`

extend path edges

On-the-fly ESG


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a

0   p

Call-Flow

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`

`return p`

Normal-Flow

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`
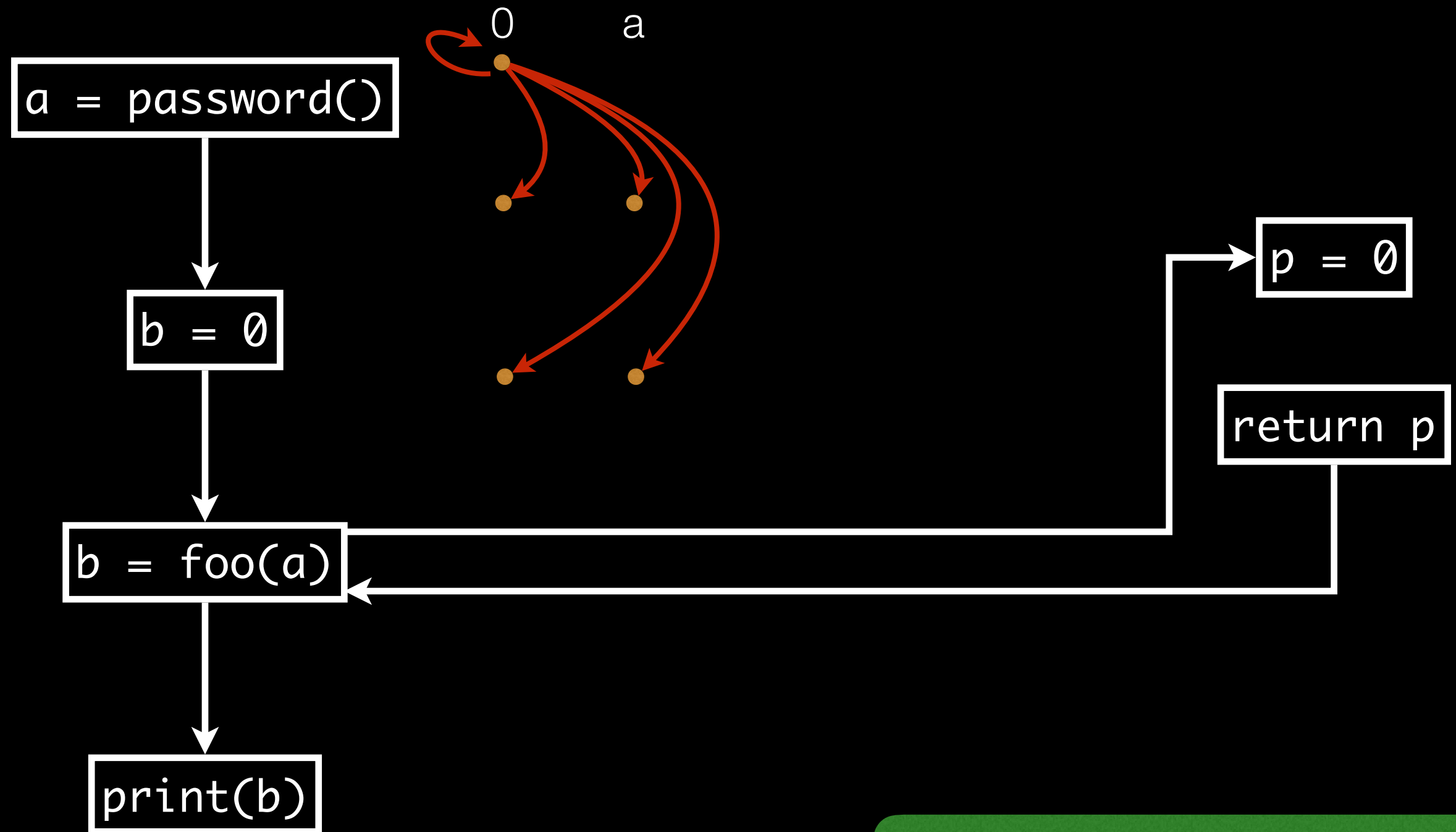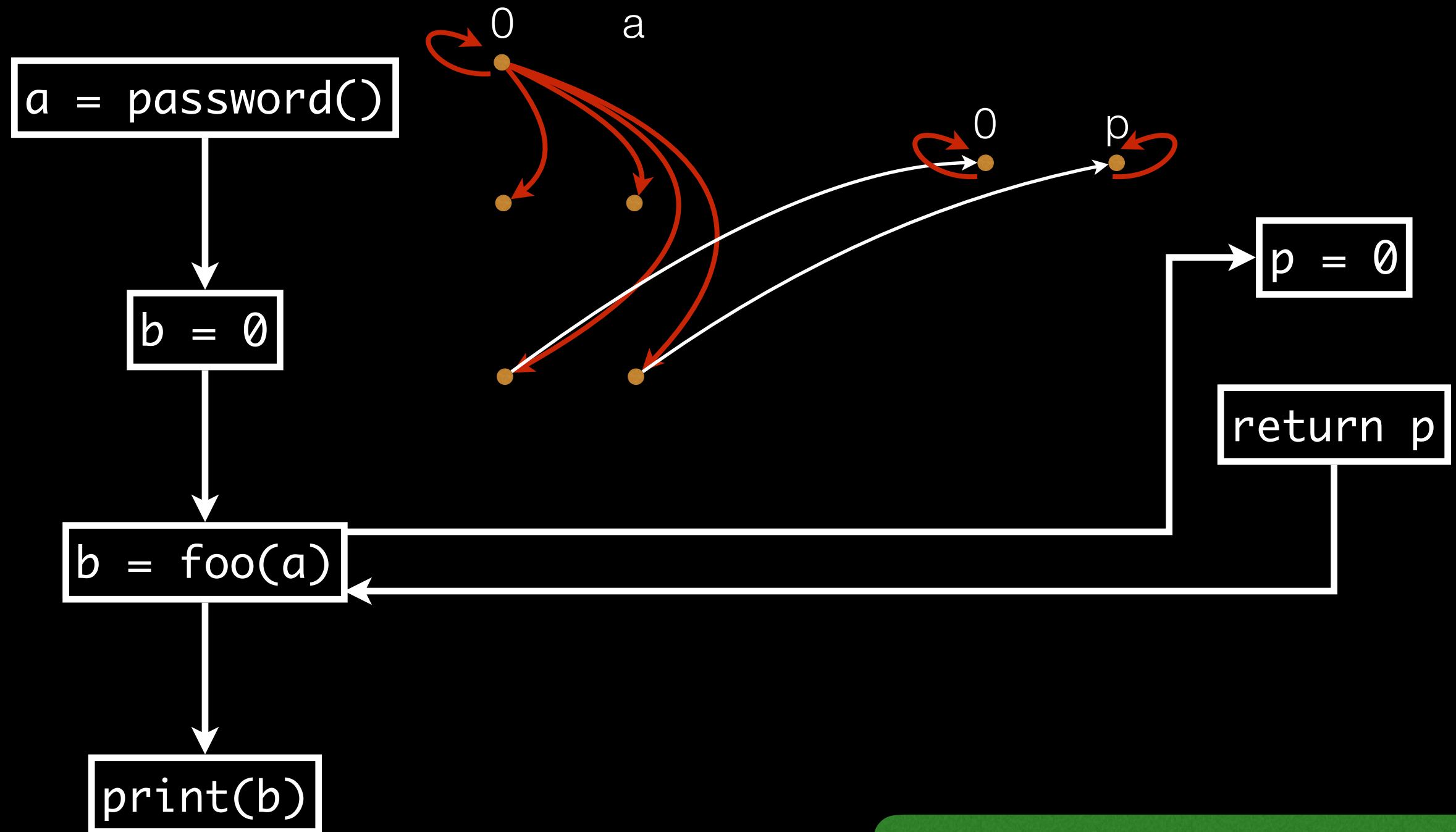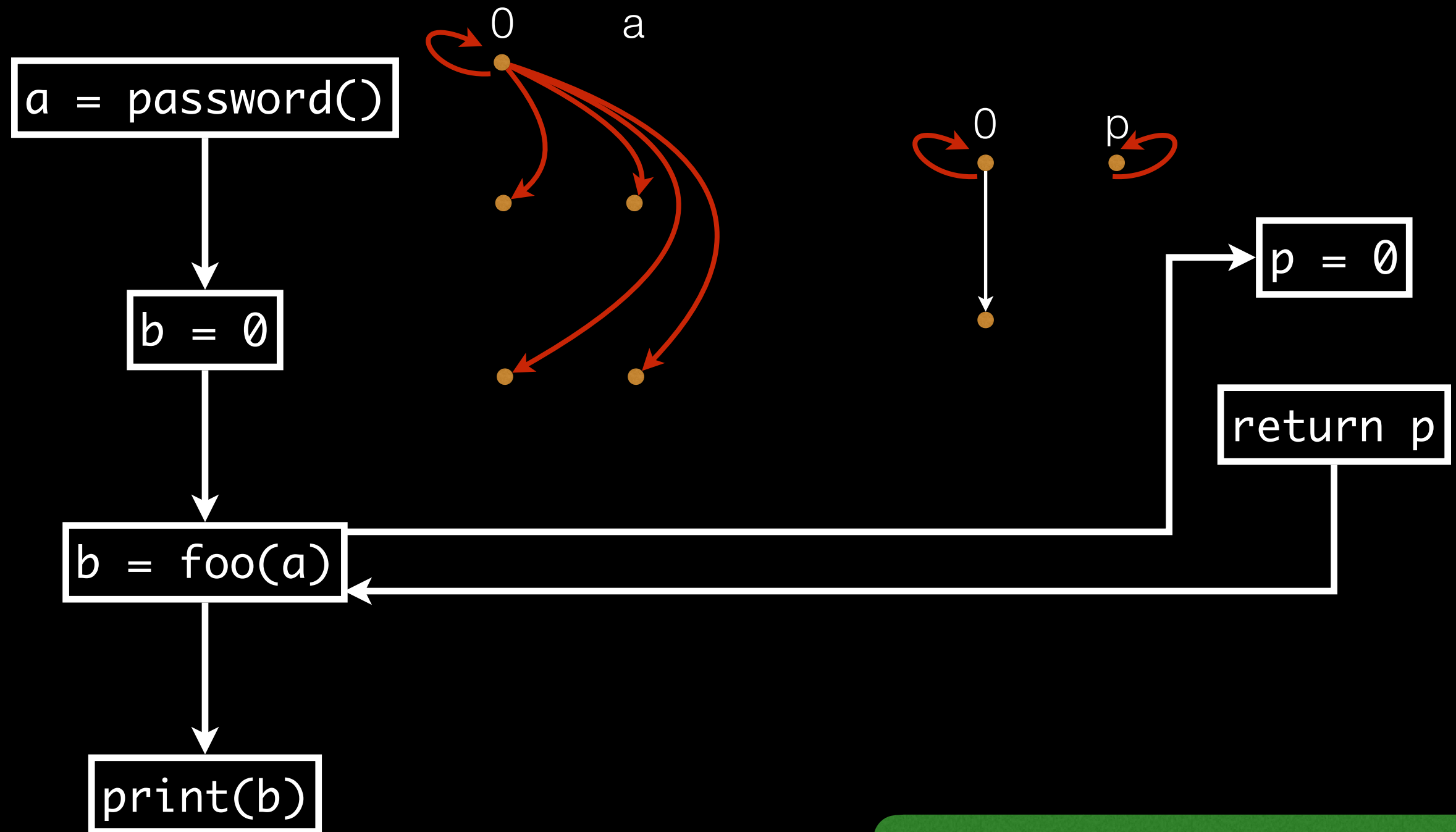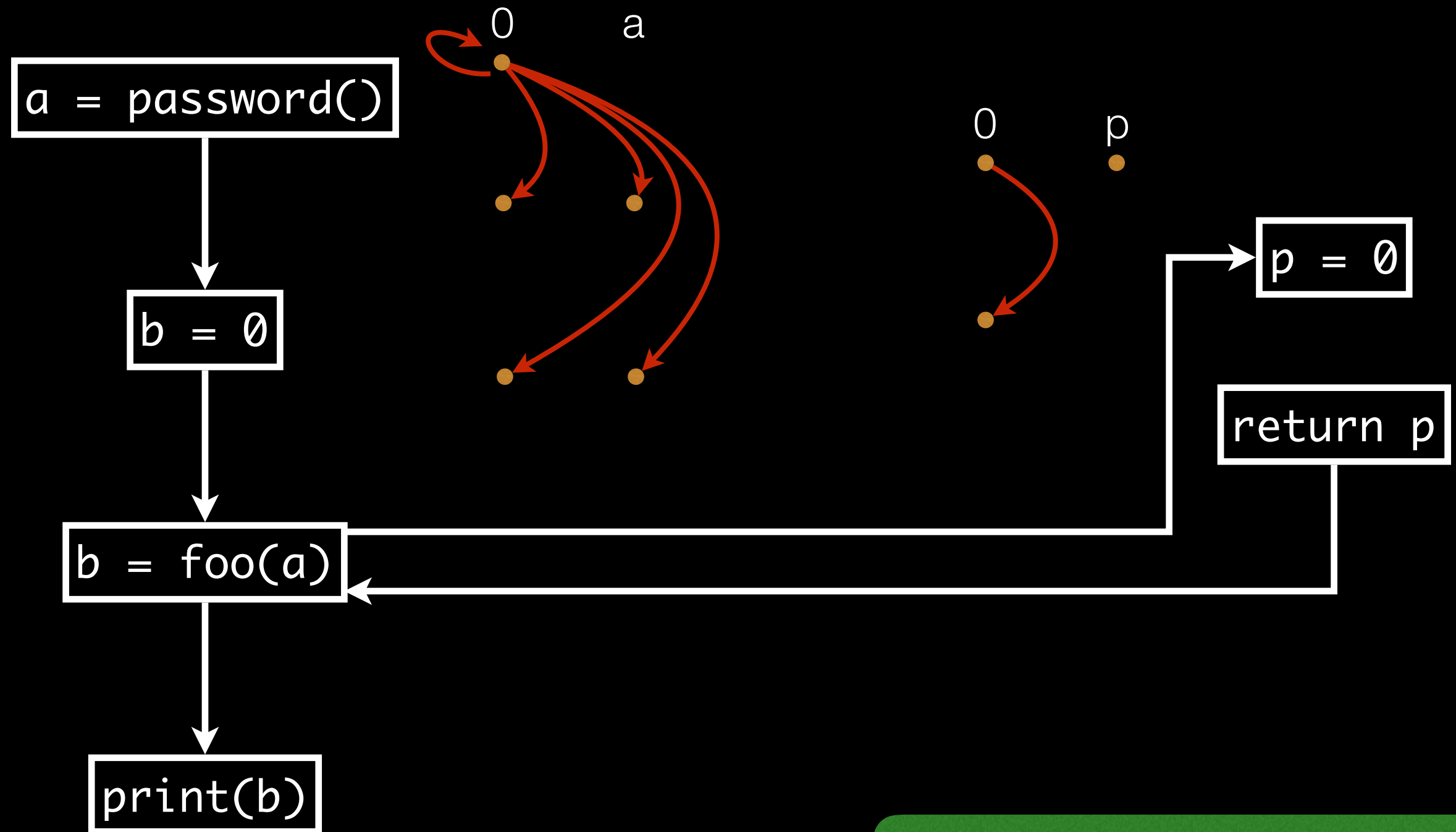
`return p`

Method Summary

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`
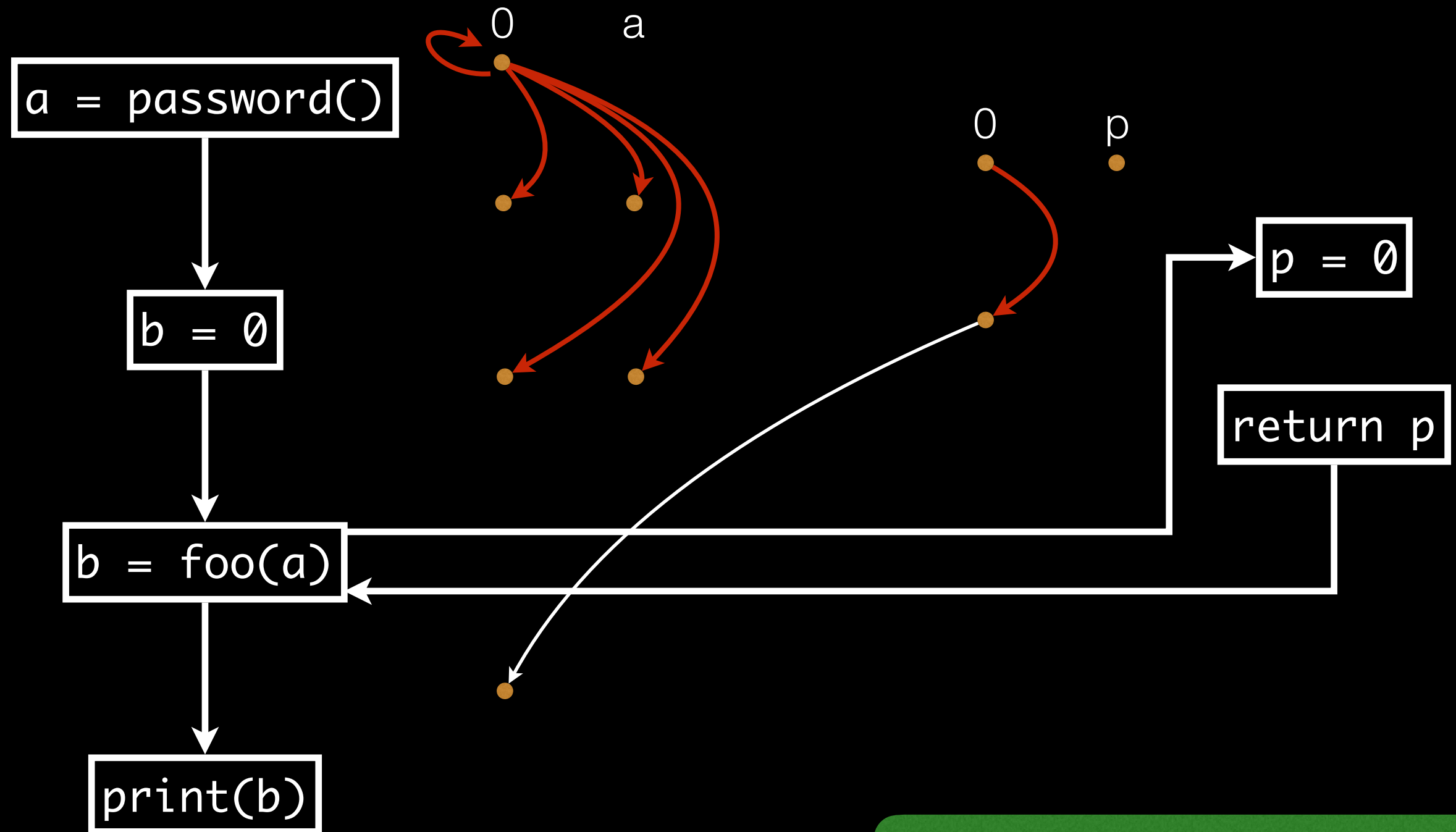
`return p`

Return-Flow

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`
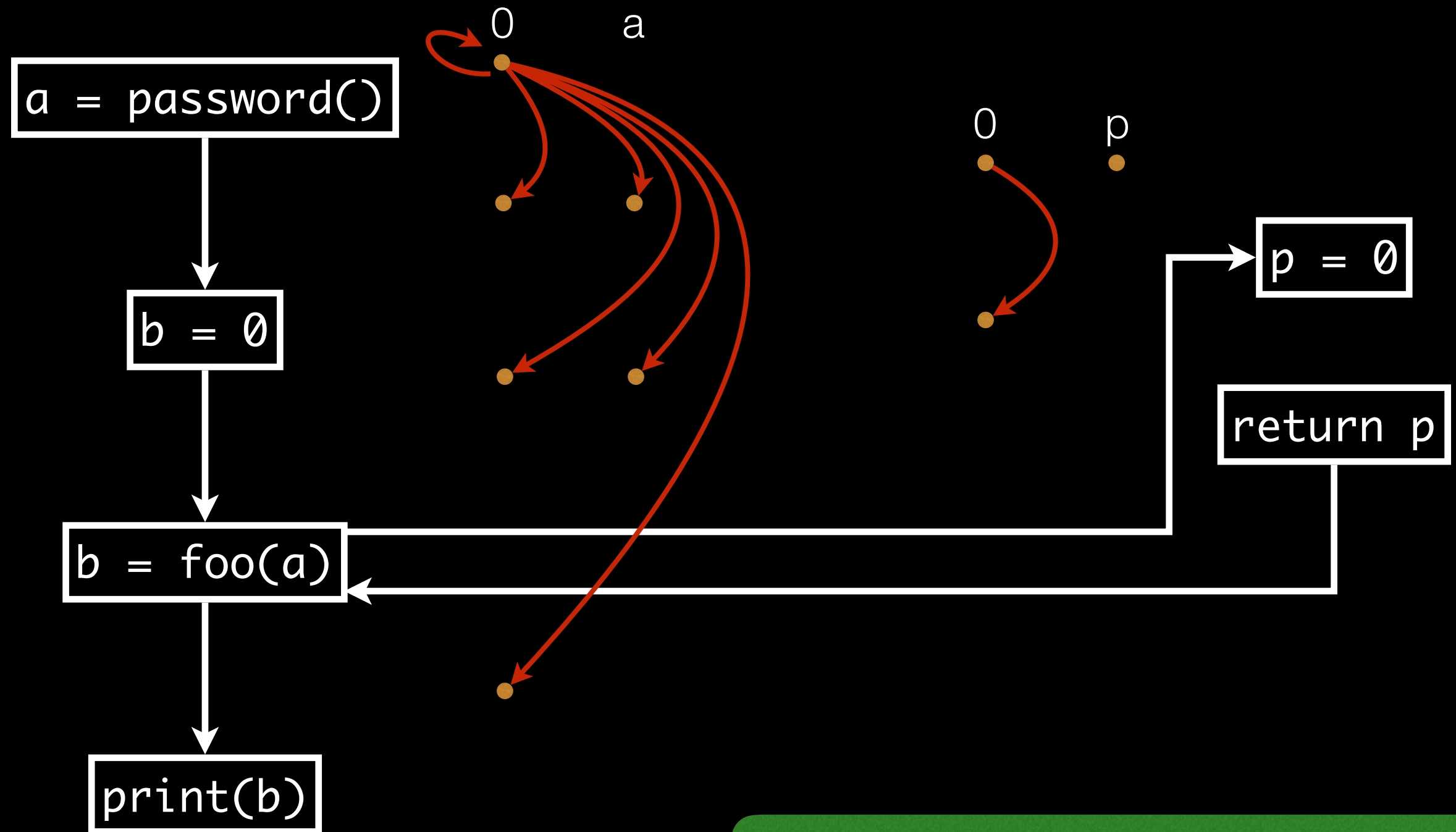
`return p`

extend path edges in caller

On-the-fly ESG


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a

0   p
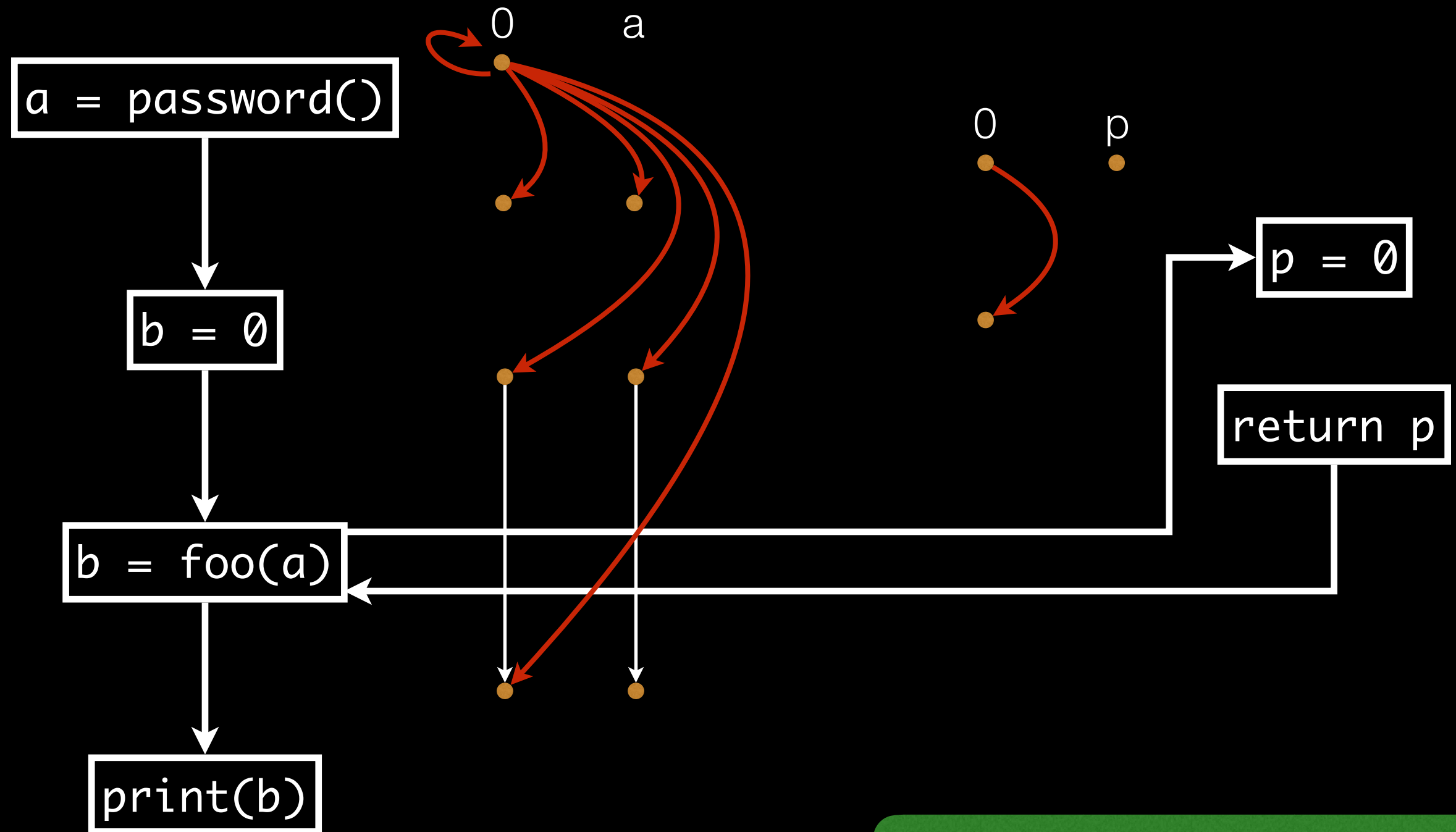
Call-to-return

On-the-fly ESG

`a = password()`

`b = 0`

`b = foo(a)`

`print(b)`

`p = 0`

`return p`

extend path edges

On-the-fly ESG


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a

0   p

Normal-Flow

On-the-fly ESG


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a

0   p

extend path edges

On-the-fly ESG


```
a = password()
```

```
b = 0
```

```
b = foo(a)
```

```
print(b)
```

```
p = 0
```

```
return p
```

0   a   b

0   p

edges never computed

@karimhamdanali   46

Tidbits!

Field-Sensitivity

a.f

Field-Sensitive

a.*

Field-Insensitive
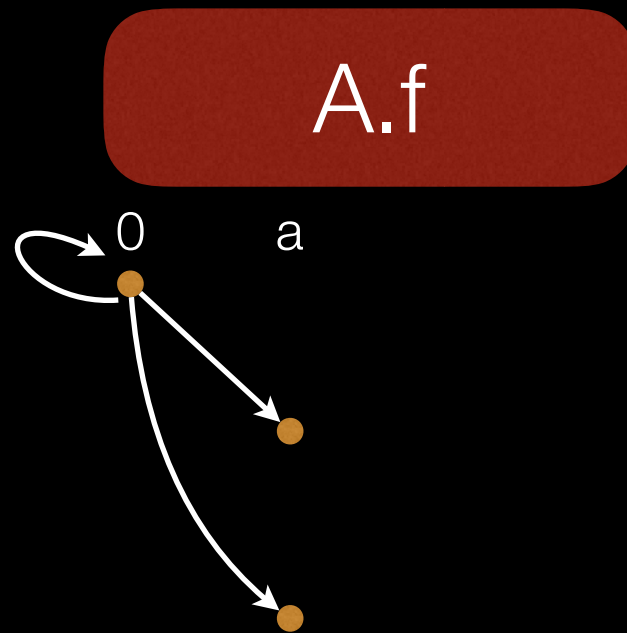
A.f

Field-Based

Field-Based

A.f

0
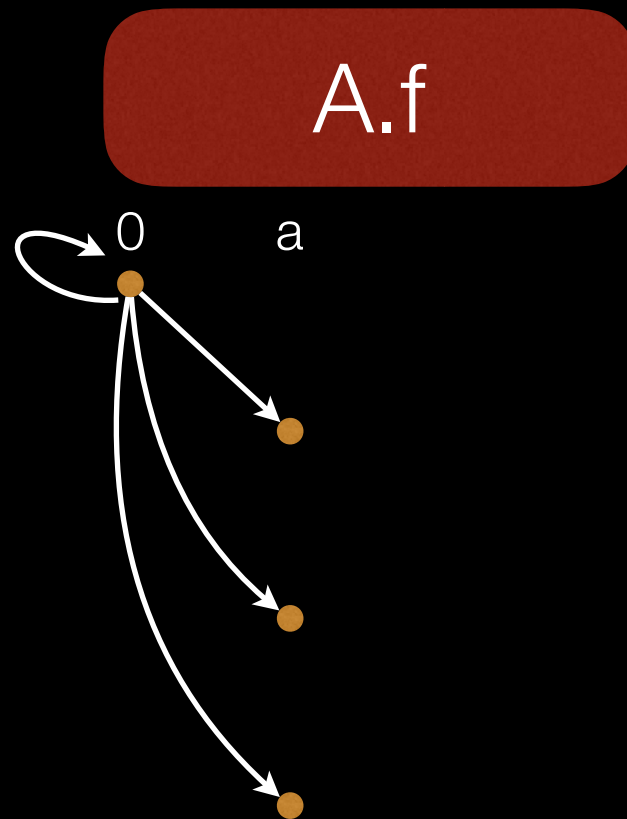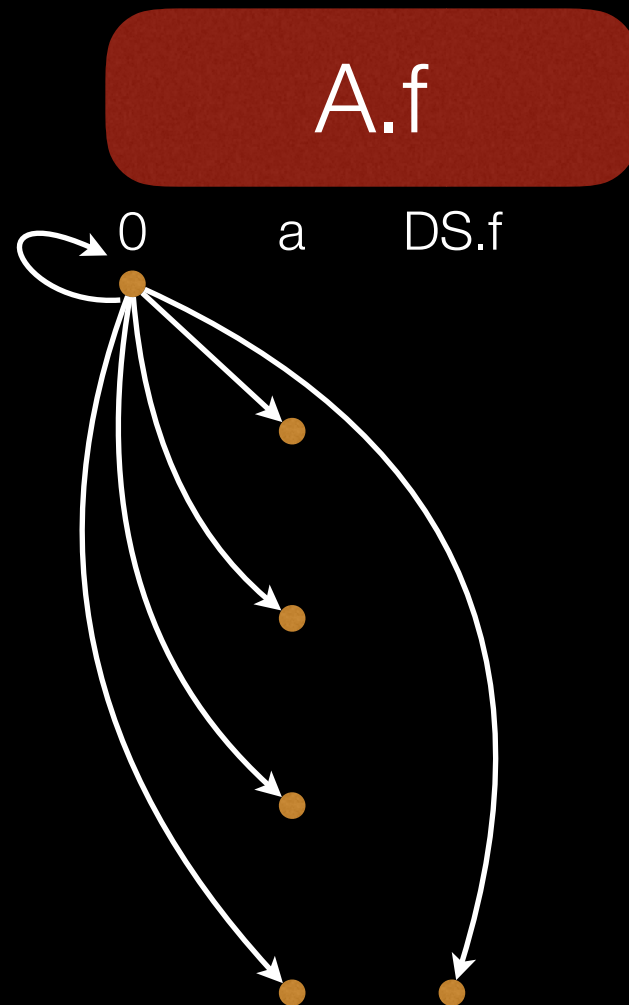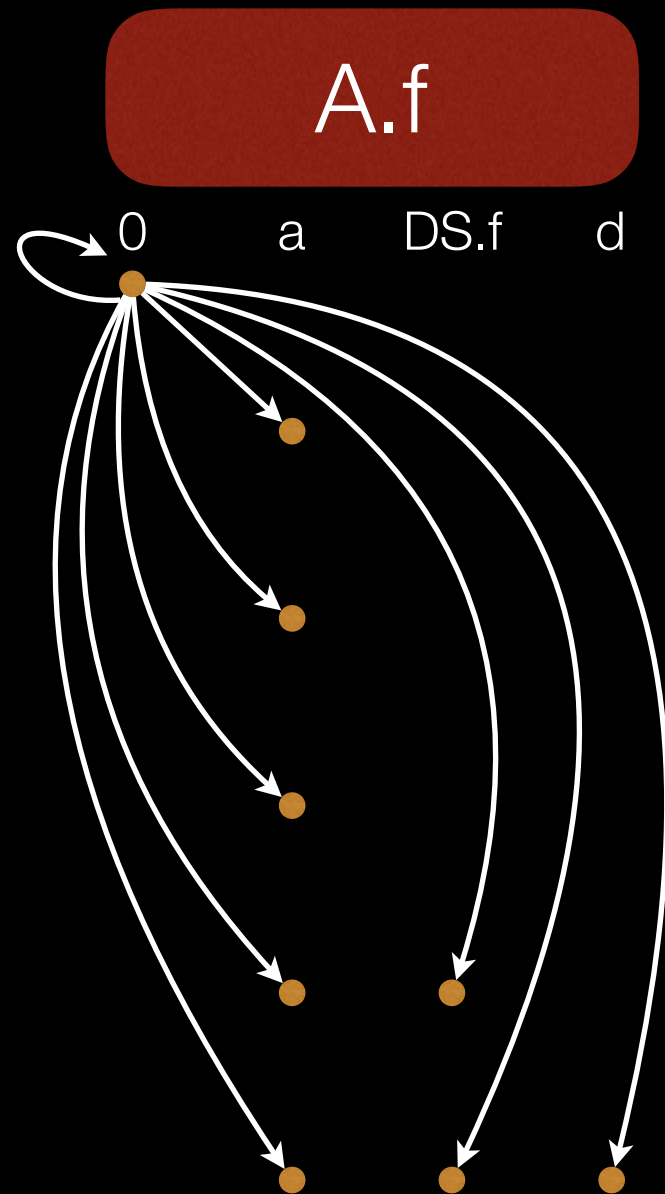
```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```

D = Locals ∪ Fields

Field-Based

A.f

0   a

```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```

D = Locals ∪ Fields

Field-Based

A.f

0   a

```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```

D = Locals ∪ Fields

Field-Based
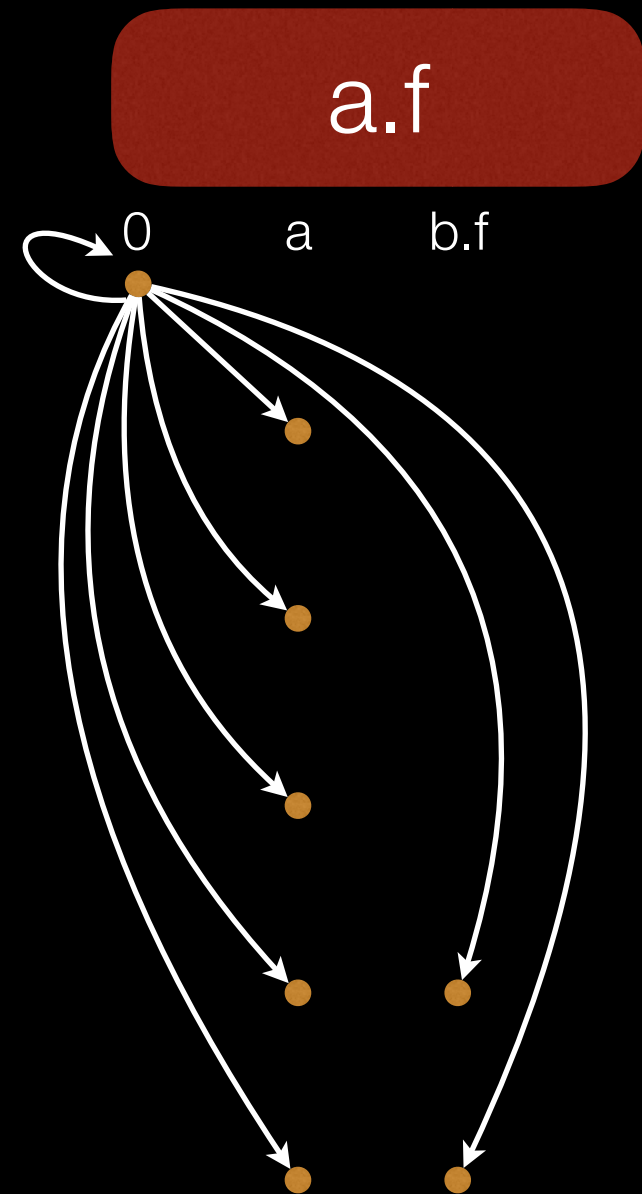
A.f

0   a

`a = source();`

`b = new DS();`

`c = new DS();`

`b.f = a;`

`d = c.f`

`sink(d);`

D = Locals ∪ Fields

Field-Based

A.f

0   a   DS.f

`a = source();`

`b = new DS();`

`c = new DS();`

`b.f = a;`

`d = c.f`

`sink(d);`

D = Locals ∪ Fields

Field-Based

A.f

0   a   DS.f   d

```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```
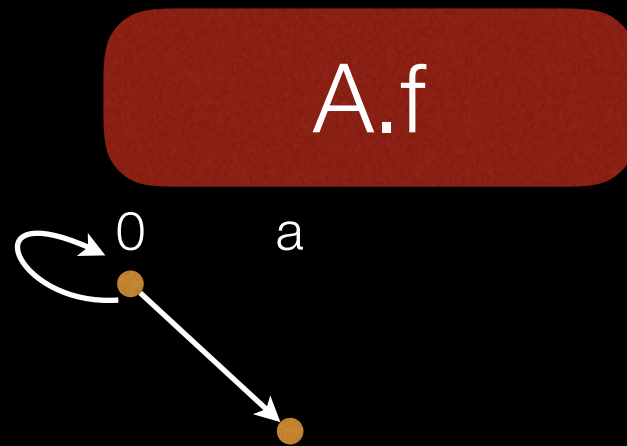
D = Locals ∪ Fields

Field-Sensitive

A.f
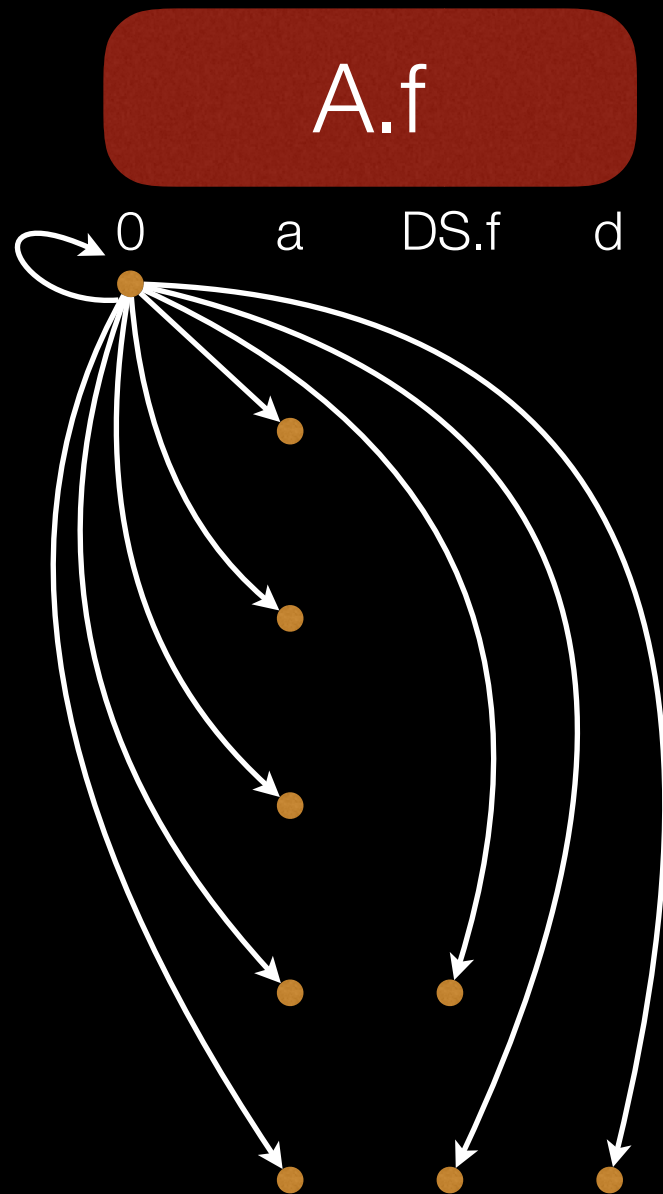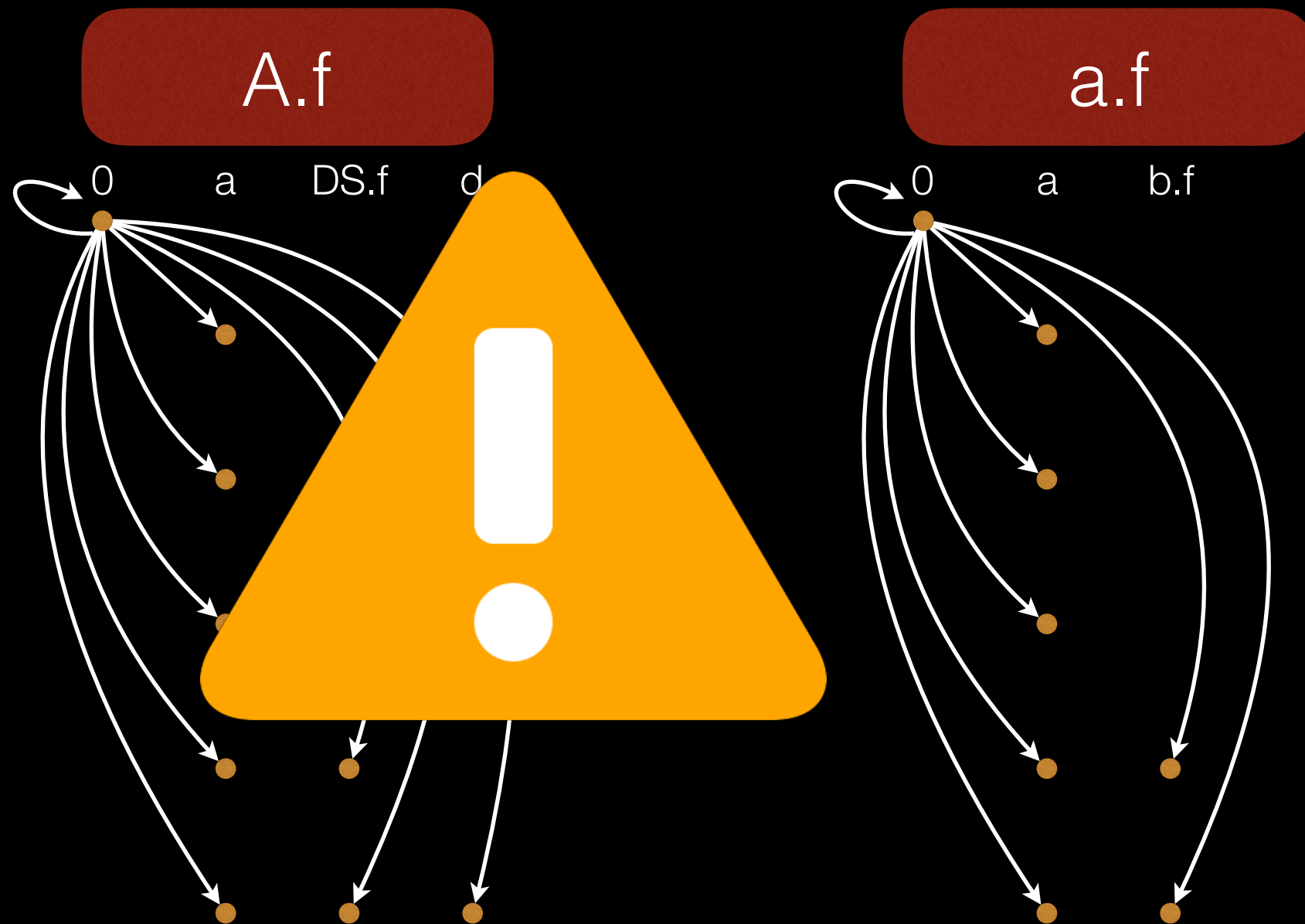
0   a   DS.f   d

```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```

a.f

0   a   b.f

$D = \{l.f_1.f_2.f_3.\ \ldots\ .f_n\}$
$l \in Locals,\ f_i \in fields$
$n \geq 0$

Field-Sensitive

A.f

0   a   DS.f   d

a.f

0   a   b.f

```
a = source();

b = new DS();

c = new DS();

b.f = a;

d = c.f

sink(d);
```
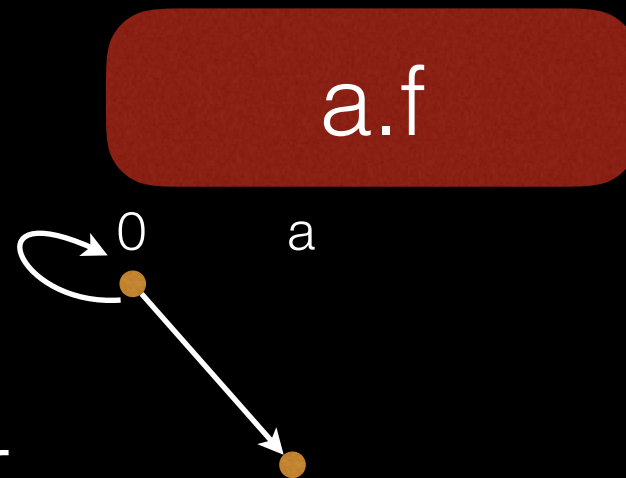
Infinite domain!!

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
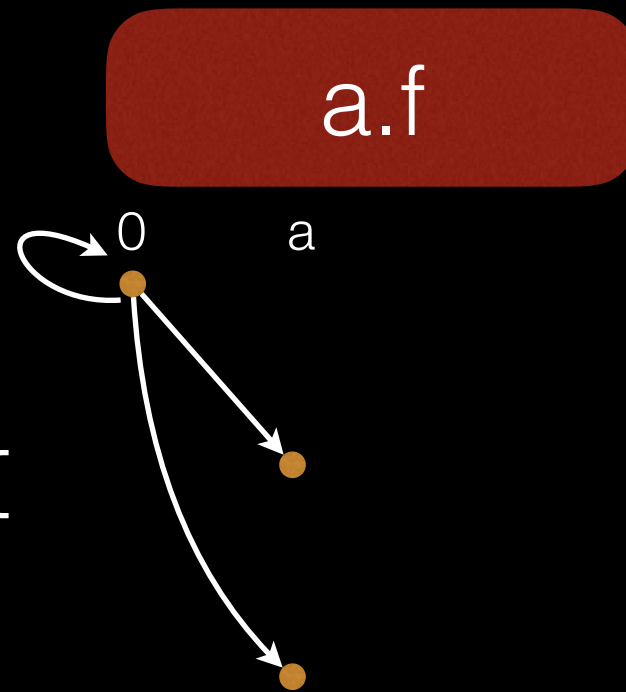$n \geq 0$

Field-Sensitive

a.f

0   a
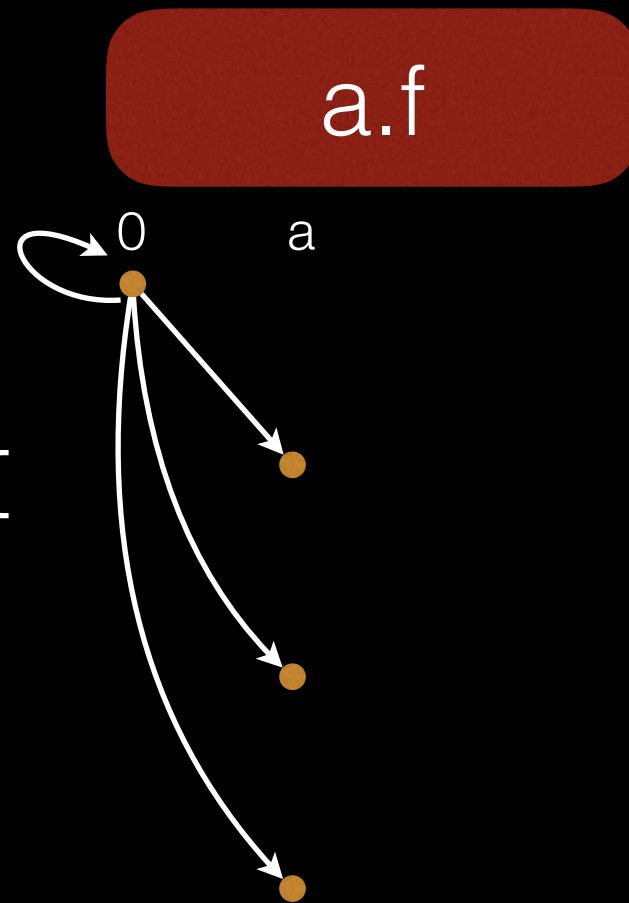
```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
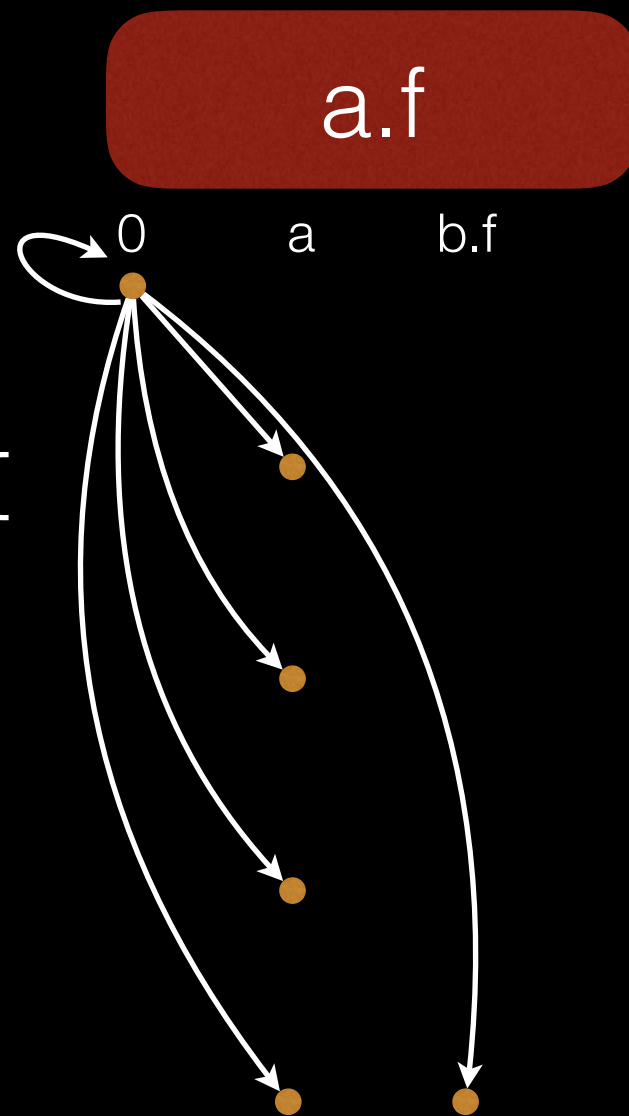$n \geq 0$

Field-Sensitive

a.f

0   a

```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in$ Locals, $f_i \in$ fields
$n \geq 0$

Field-Sensitive

a.f

a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$

Field-Sensitive

a.f

0   a   b.f
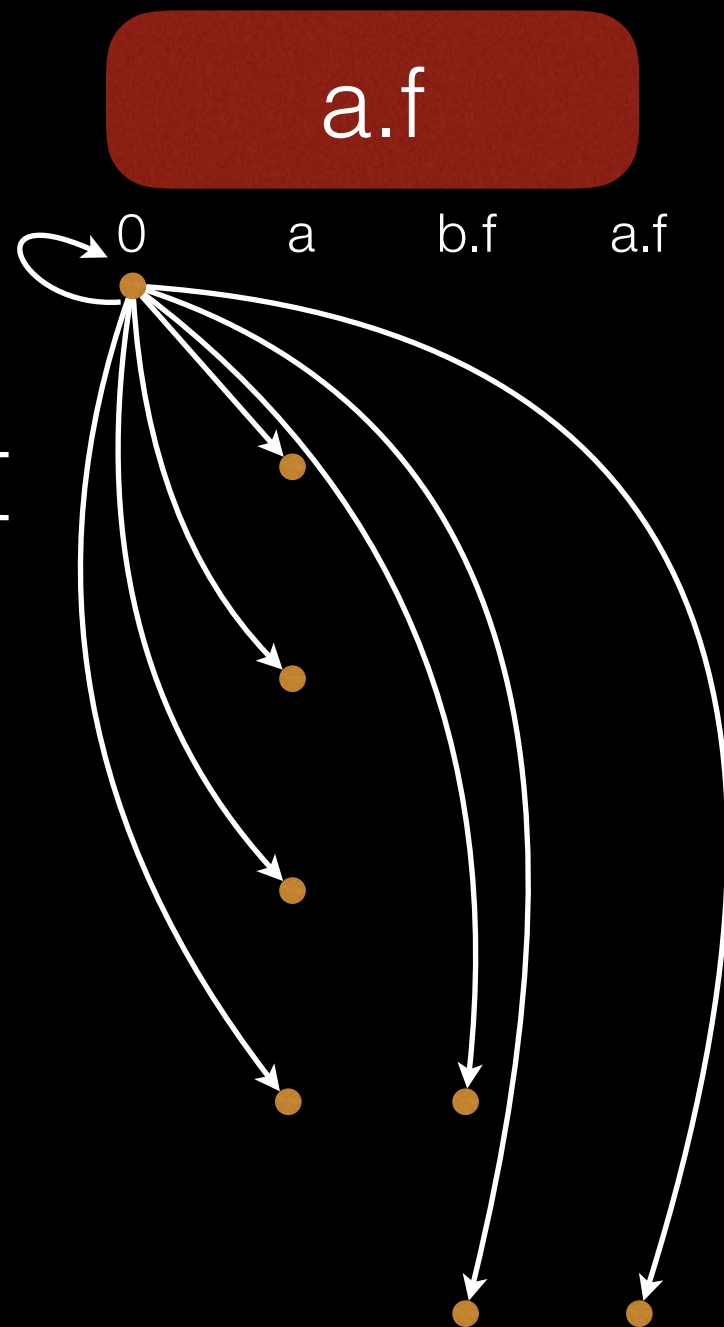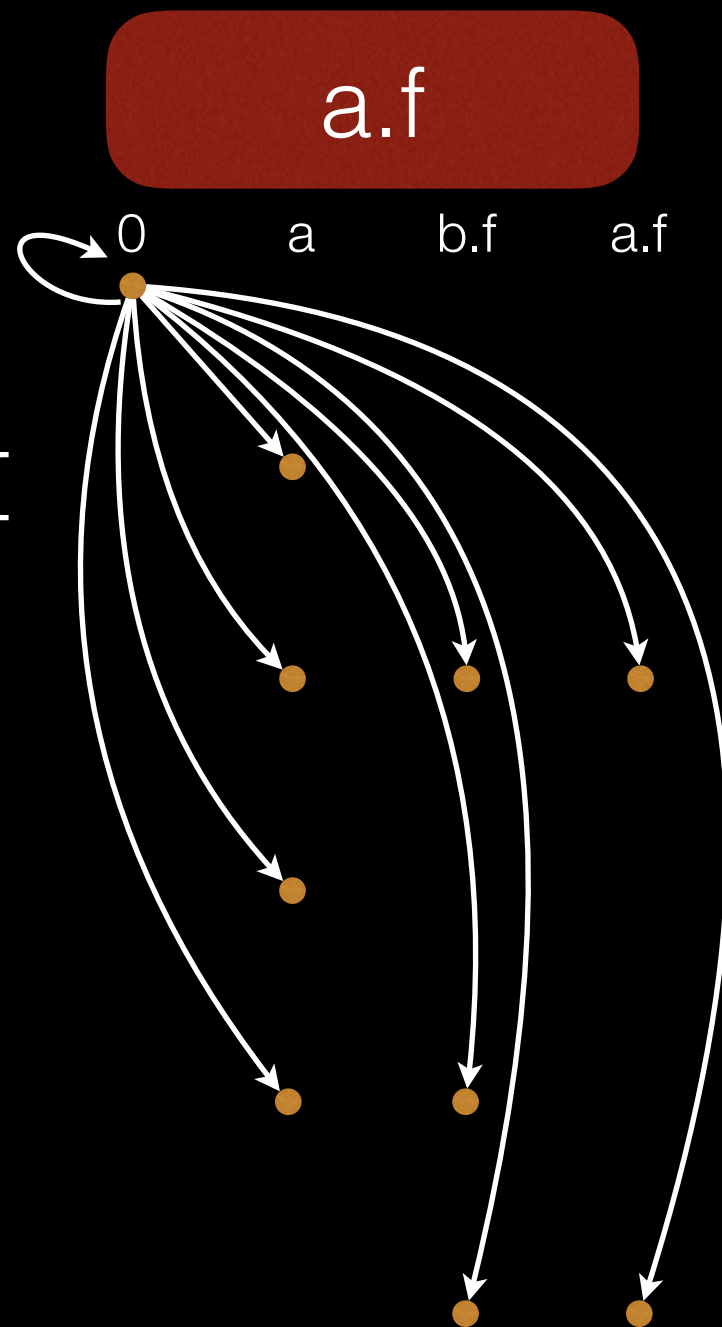
```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$
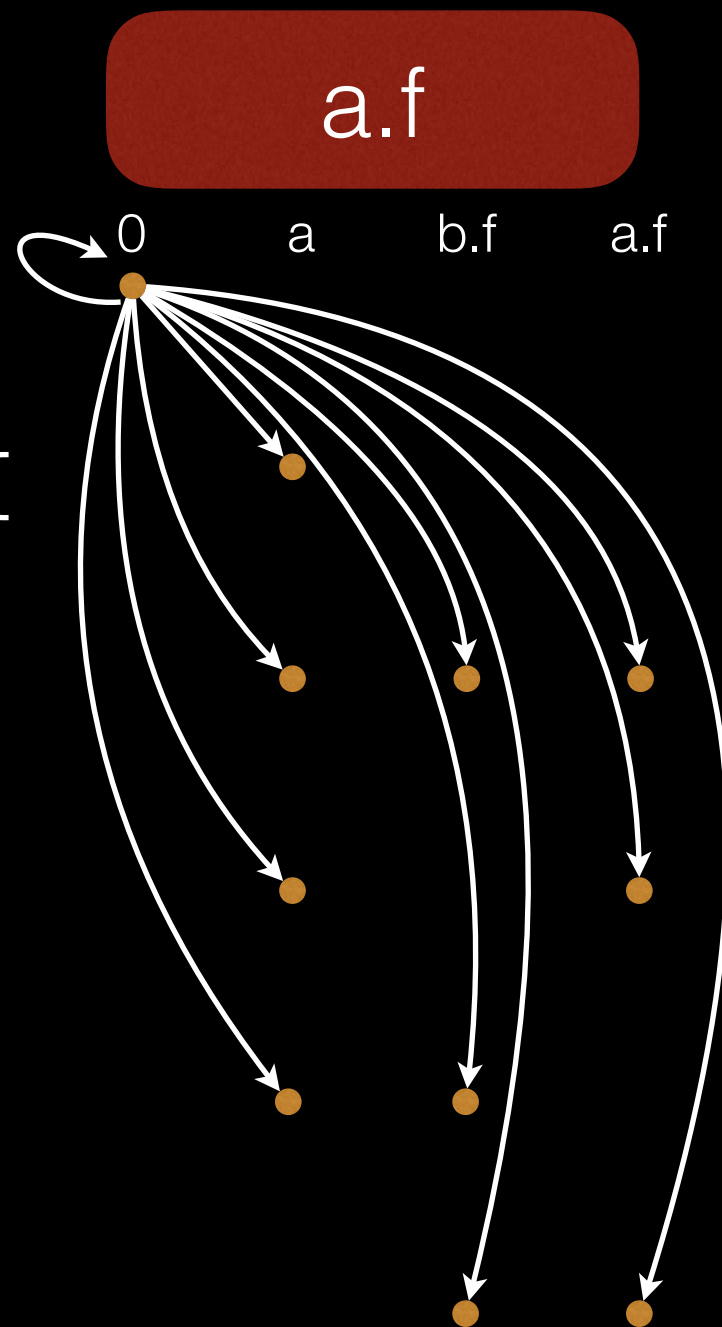
Field-Sensitive

a.f

0   a   b.f   a.f

```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$

Field-Sensitive

a.f

0   a   b.f   a.f
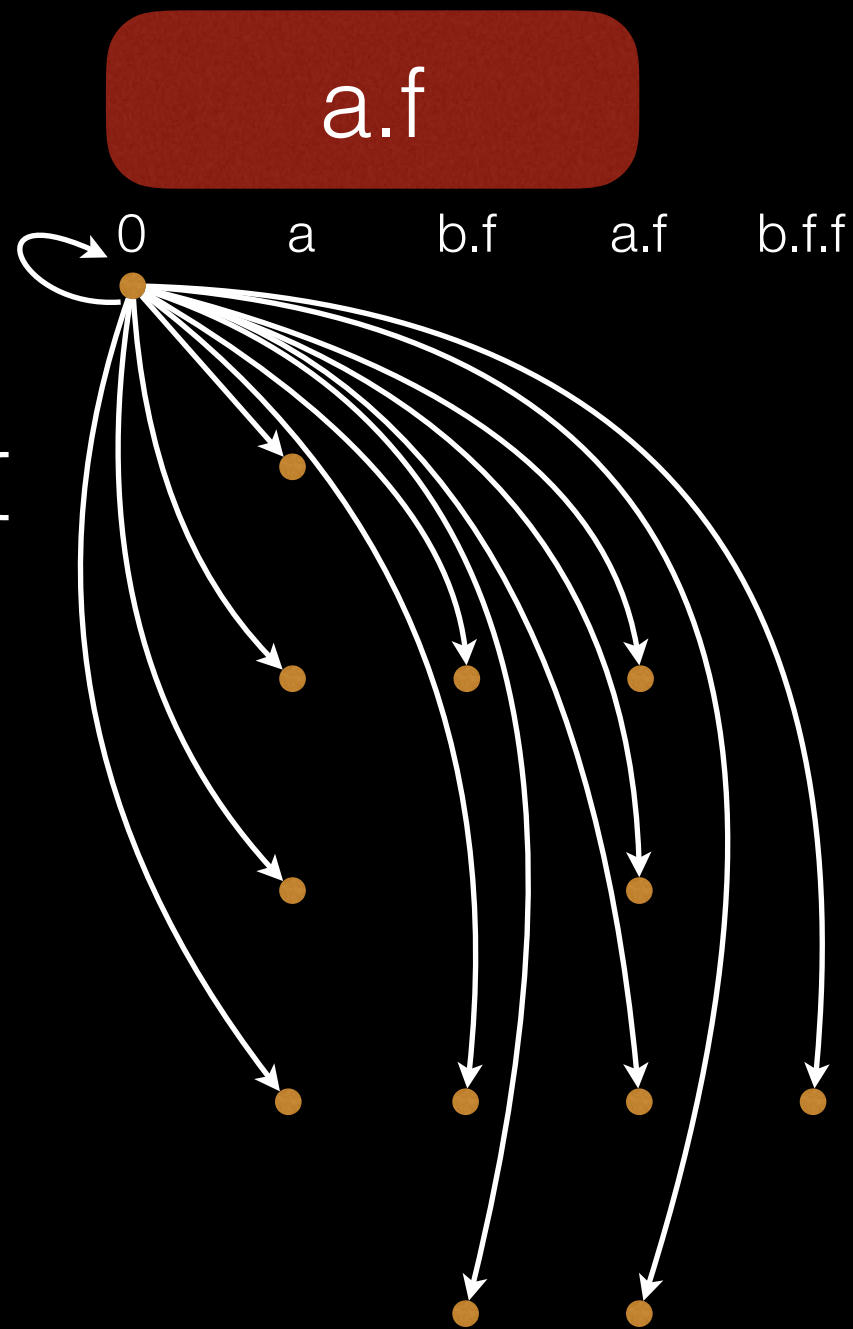
```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in$ Locals, $f_i \in$ fields
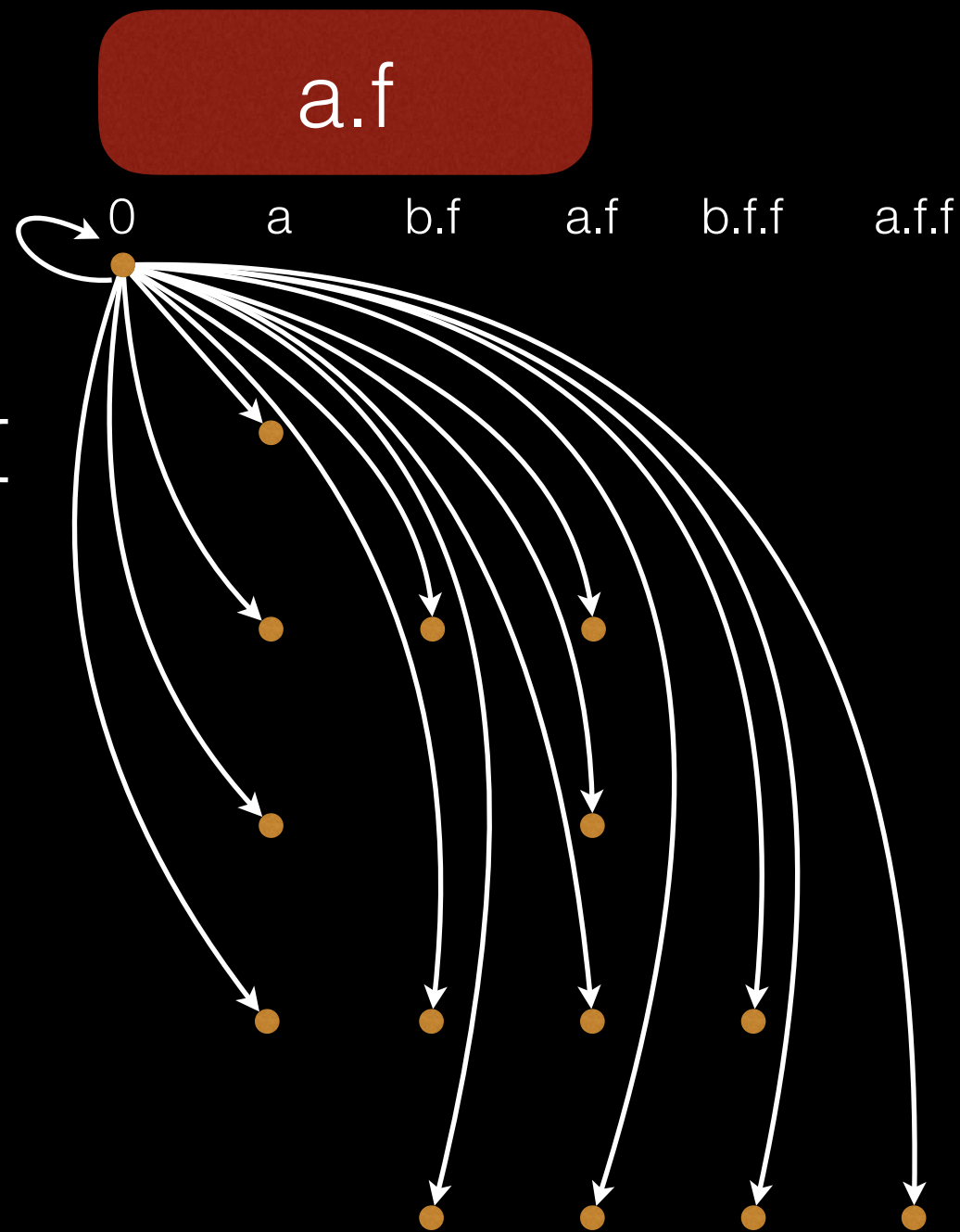$n \geq 0$

Field-Sensitive

a.f

0   a   b.f   a.f

```
a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in$ Locals, $f_i \in$ fields
$n \geq 0$

63

Field-Sensitive

a.f

0   a   b.f   a.f   b.f.f
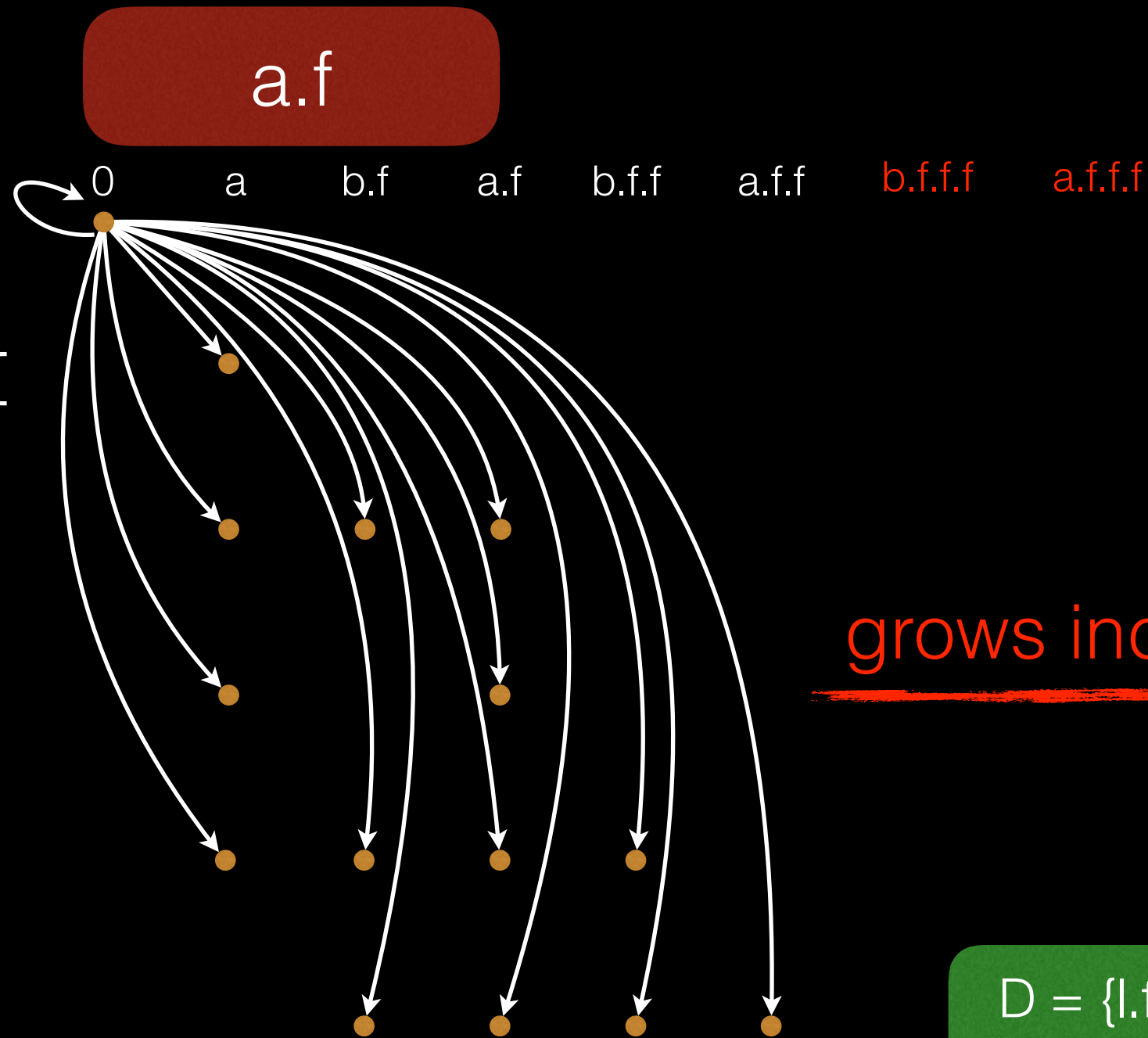
```
a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$

Field-Sensitive

a.f

0   a   b.f   a.f   b.f.f   a.f.f

```
a = source();

while(rand()) {

    b = new DS();

    b.f = a;

    a = b;
}

sink(d);
```

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$

Field-Sensitive

a.f

0   a   b.f   a.f   b.f.f   a.f.f   b.f.f.f   a.f.f.f

```
a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);
```

grows indefinitely

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$n \geq 0$

k-Limiting

a.f

0　　a　　b.f　　a.f　　b.f.f　　a.f.f　　b.f.f.f　　a.f.f.f
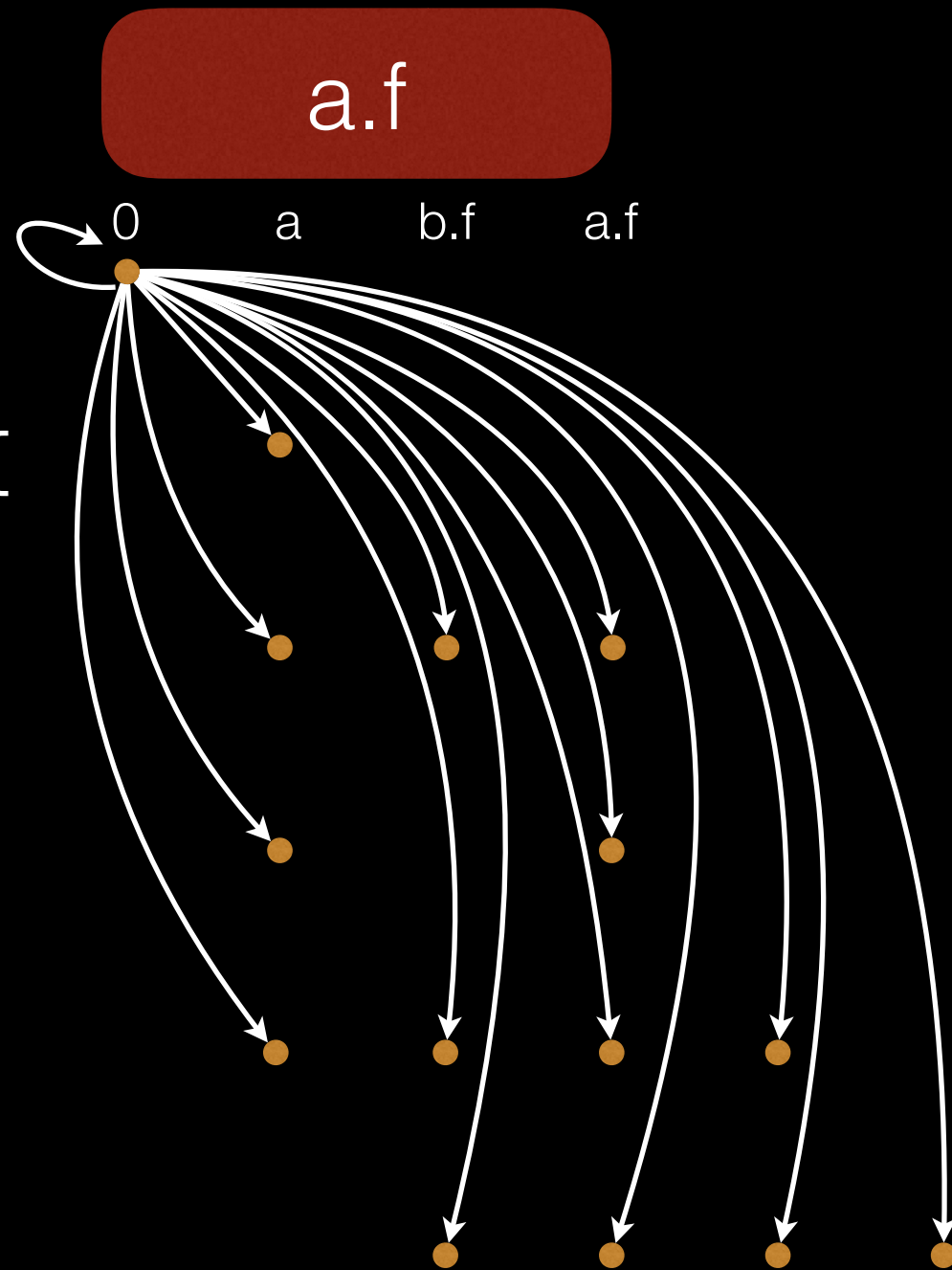
```
a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);
```
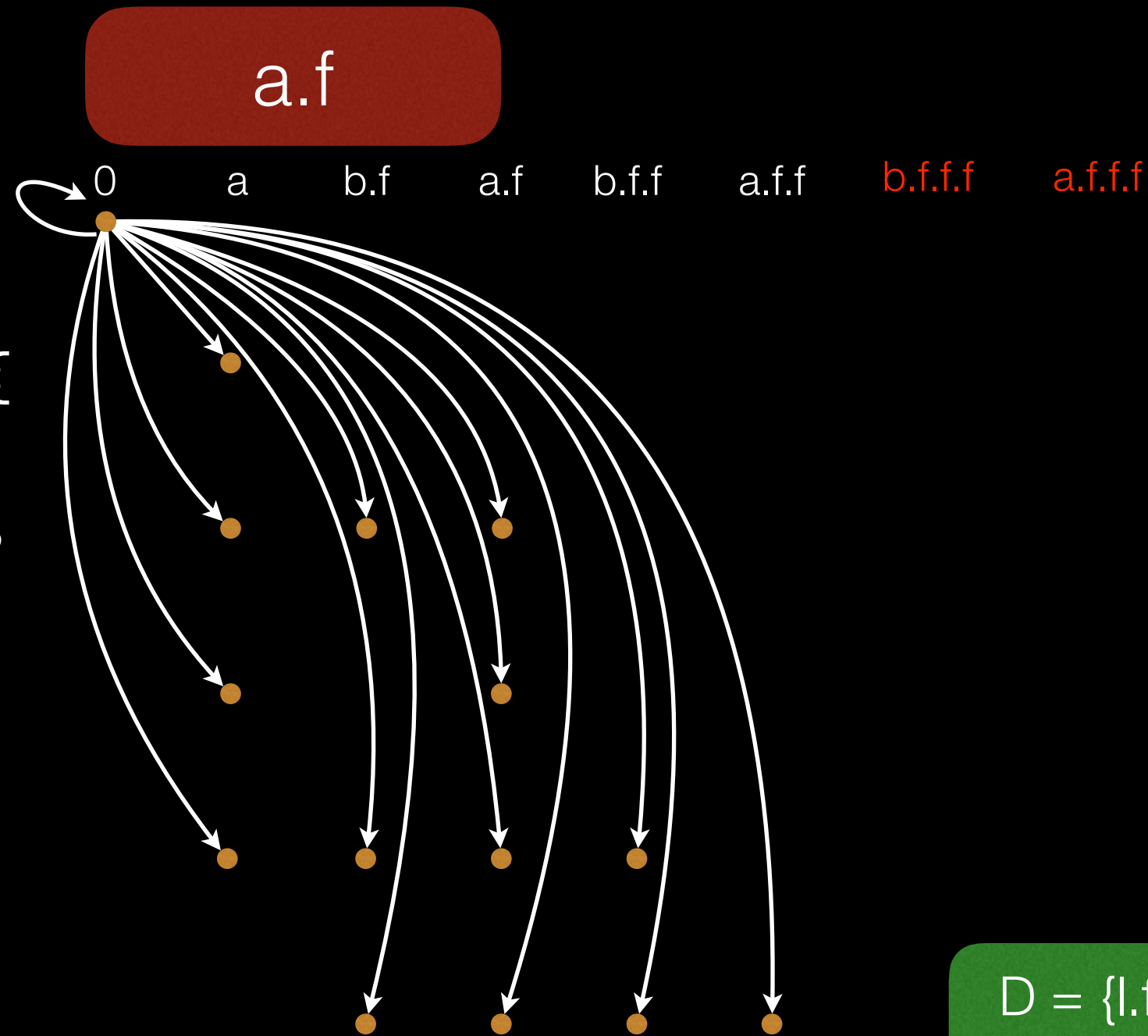
$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$0 \leq n \leq k$

k-Limiting

a.f

0   a   b.f   a.f

```
a = source();

while(rand()) {

  b = new DS();

  b.f = a;

  a = b;
}

sink(d);
```

unsound domain
beyond k => imprecise

k = 2

$D = \{l.f_1.f_2.f_3. \ldots .f_n\}$
$l \in Locals, f_i \in fields$
$0 \leq n \leq k$

k-Limiting

0   a   b.f   c.f.f   d.f   e

k ≥ 2

```
a = source();

b = new DS();

c = new DS();

b.f = a;

c.f = b;

d = c.f;

e = d.g;

sink(e);
```

k-Limiting

0   a   b.f   c.f.f   d.f   e   0   a   b.f   c.f.*   d.*   e.*

```
a = source();
```

`k ≥ 2`

```
b = new DS();

c = new DS();

b.f = a;

c.f = b;

d = c.f;

e = d.g;

sink(e);
```

`k = 1`
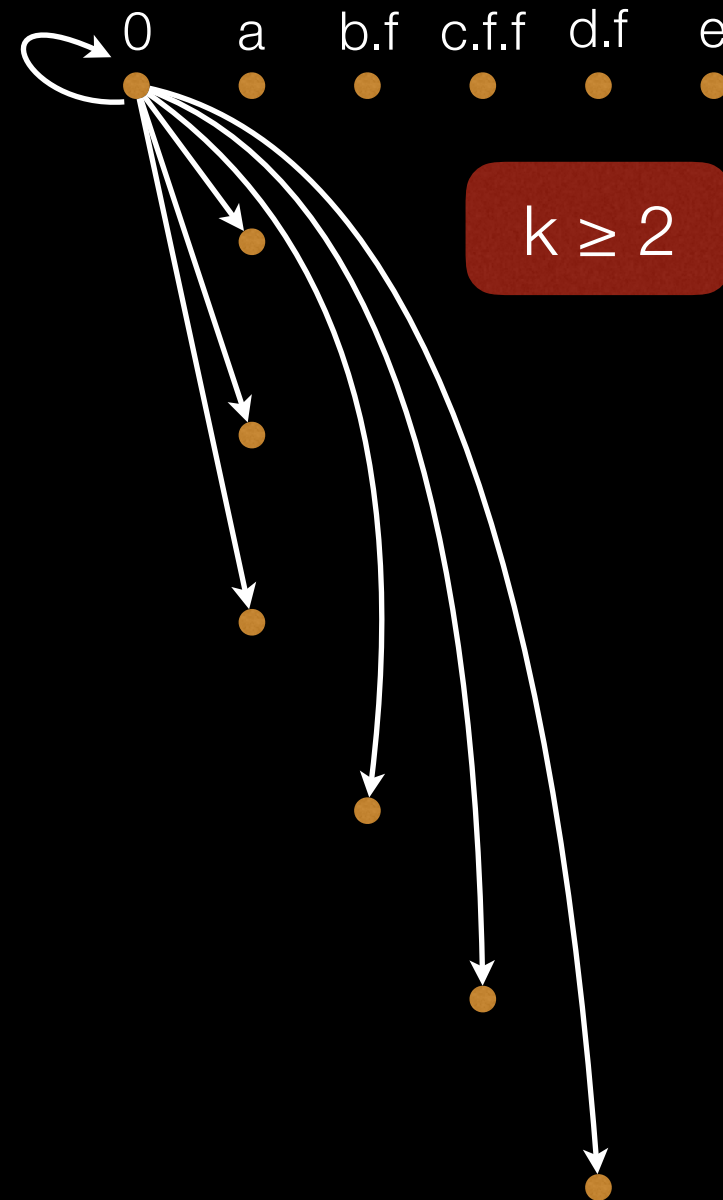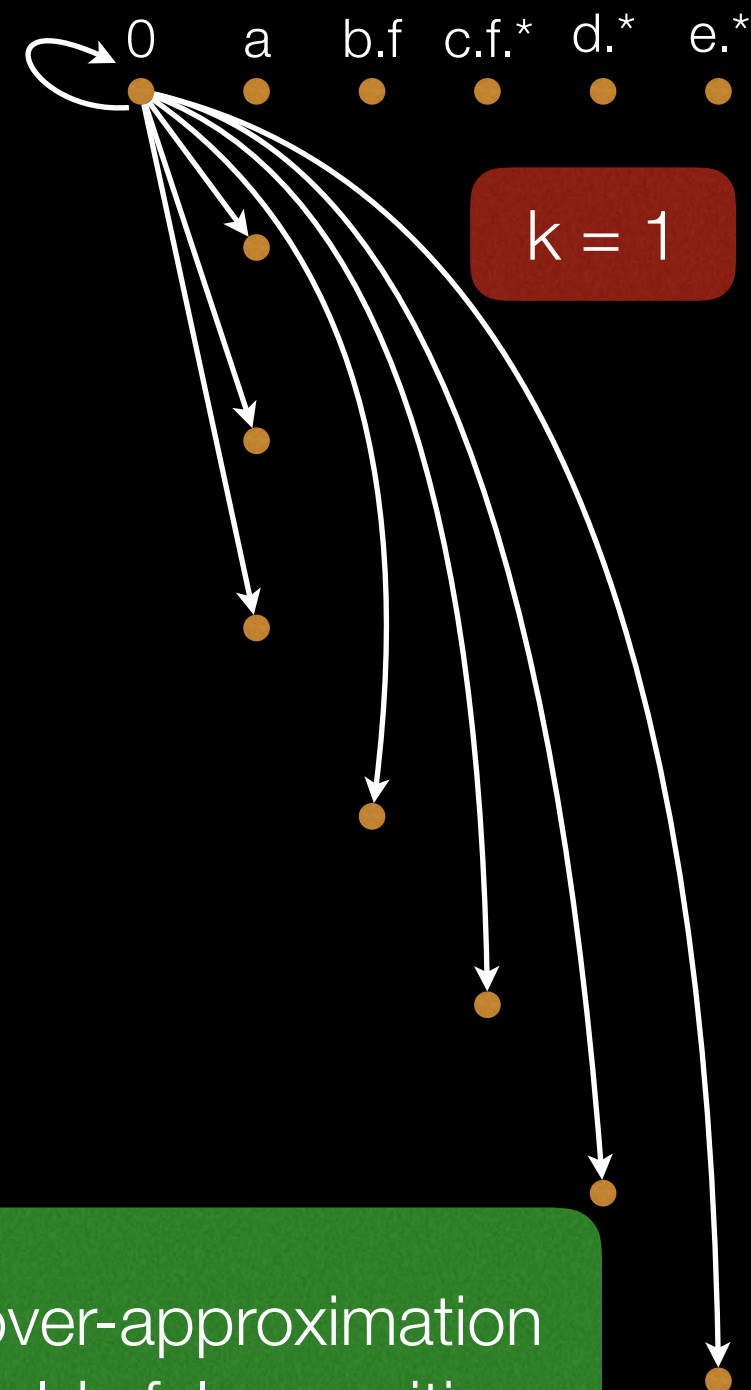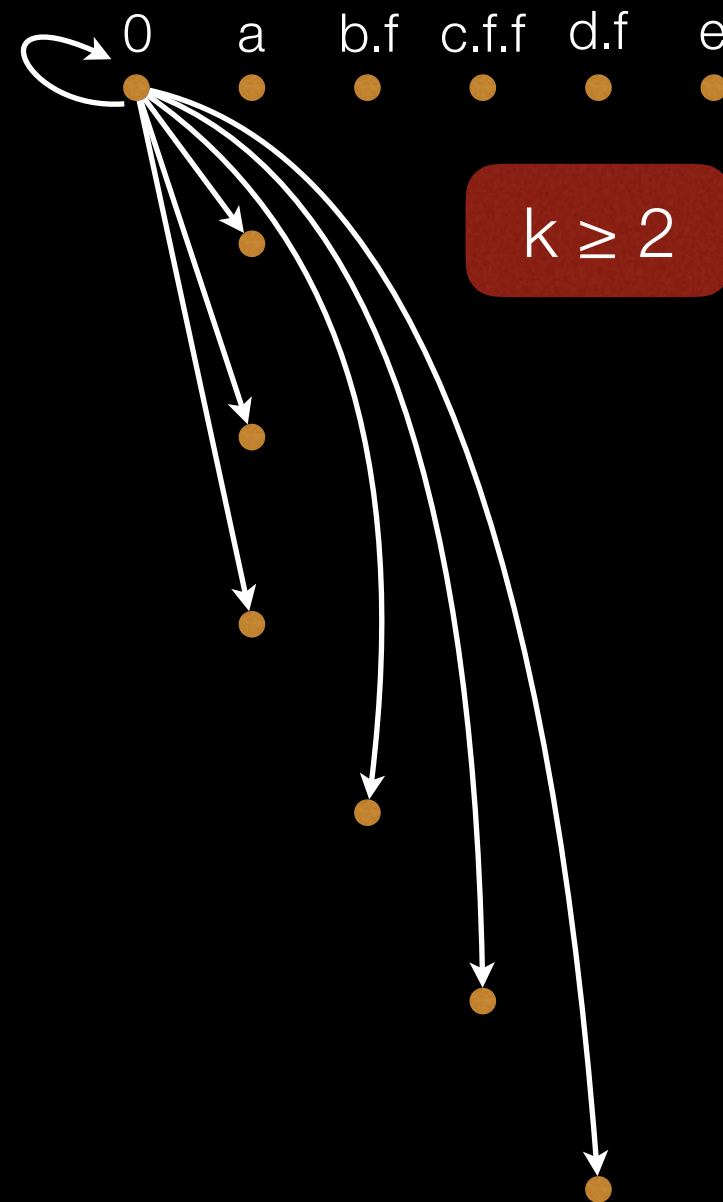
over-approximation
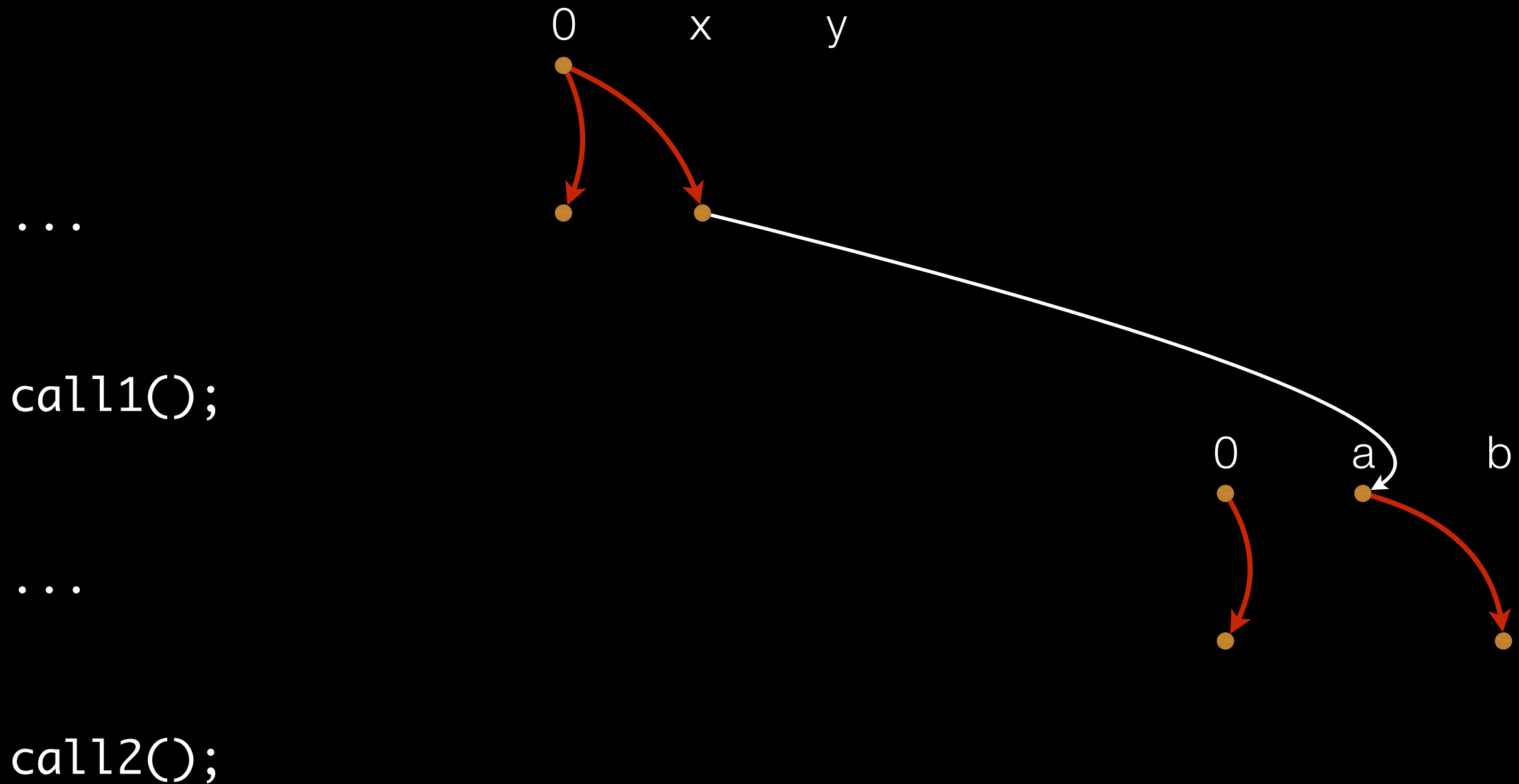yields false positives

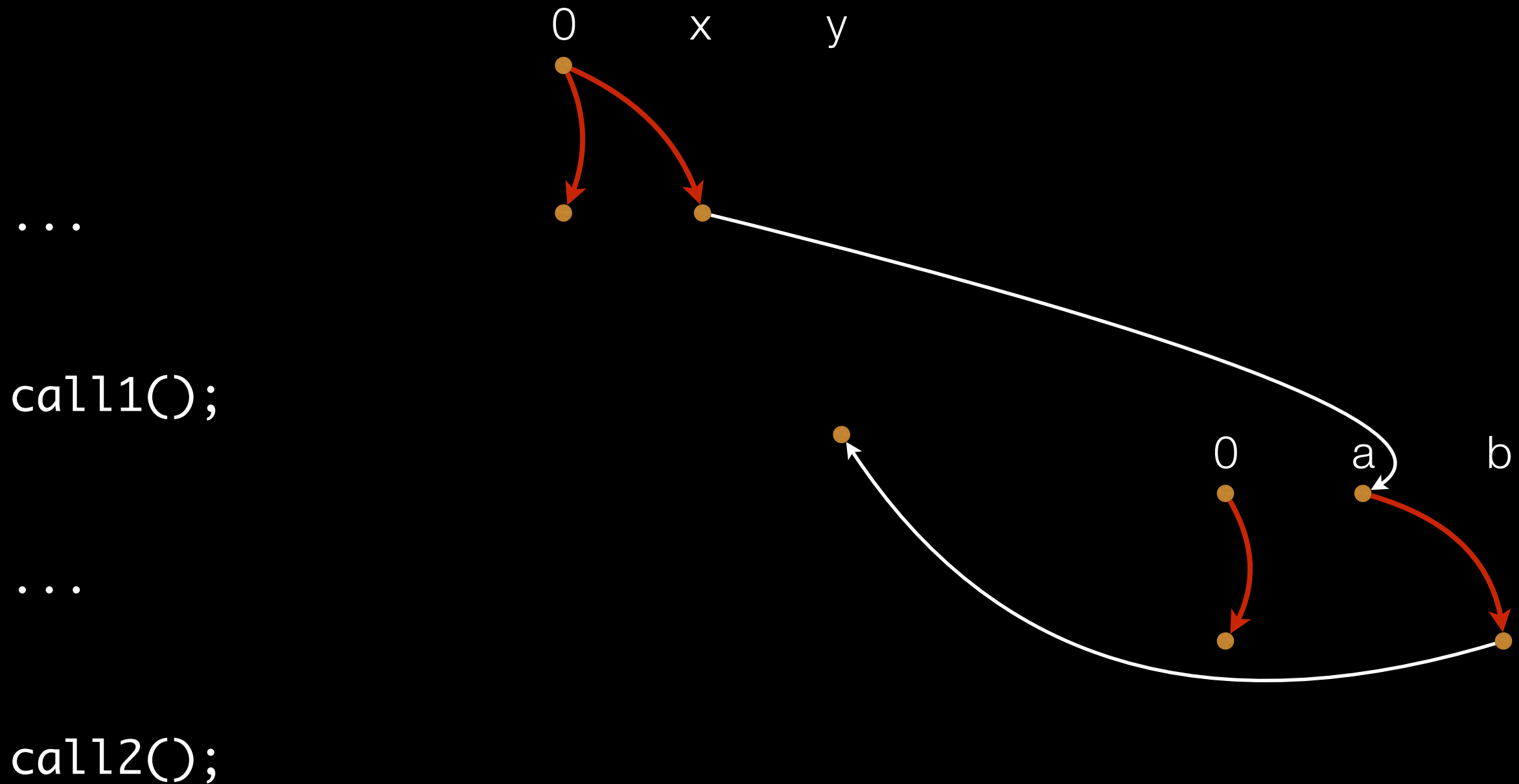# Tidbits

- Flow Sensitivity?

- Context Sensitivity?

Tidbits

- Flow Sensitivity?
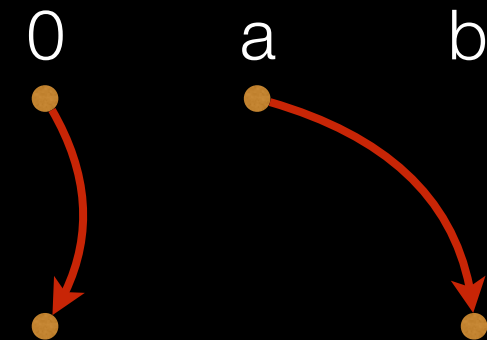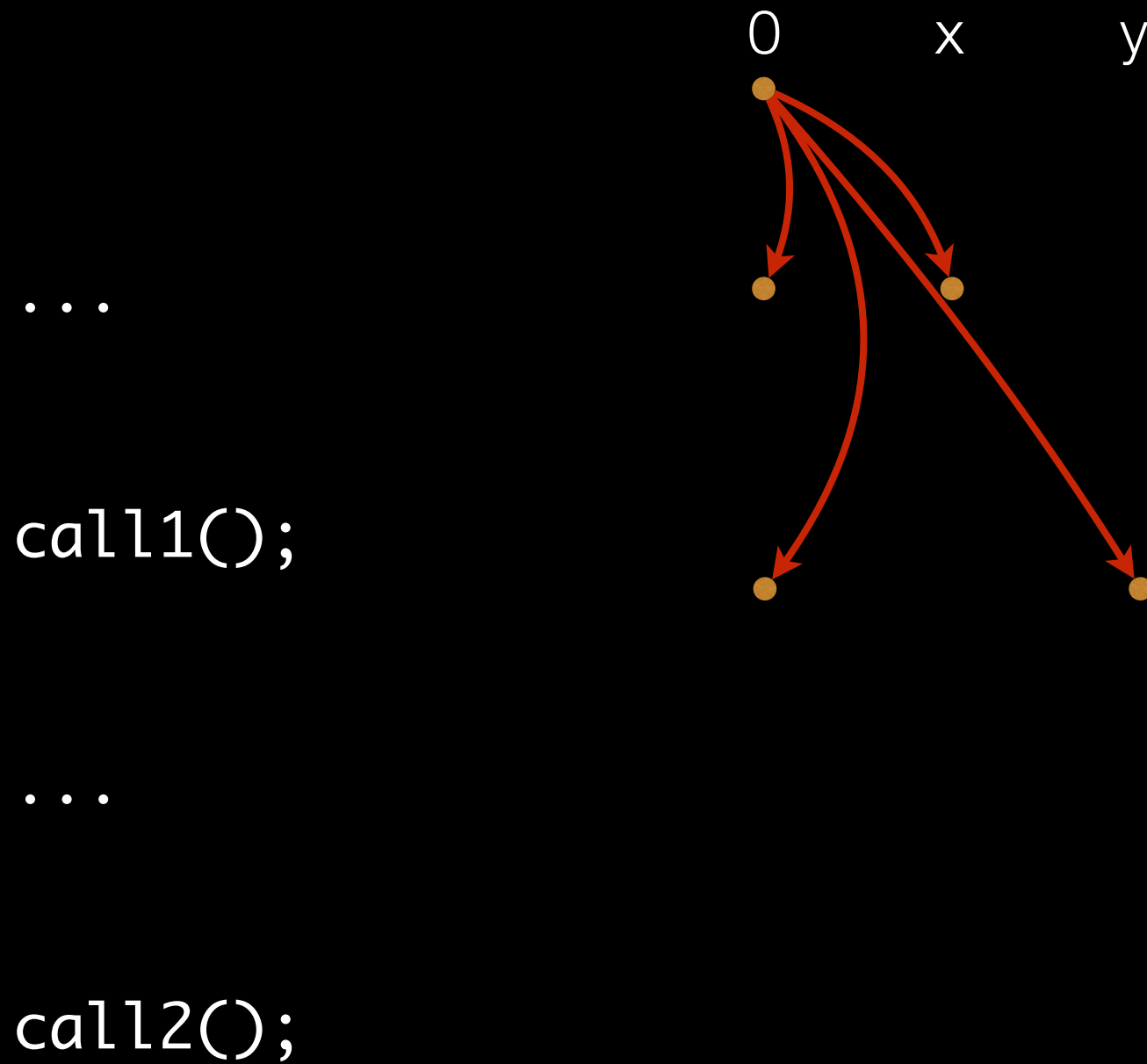
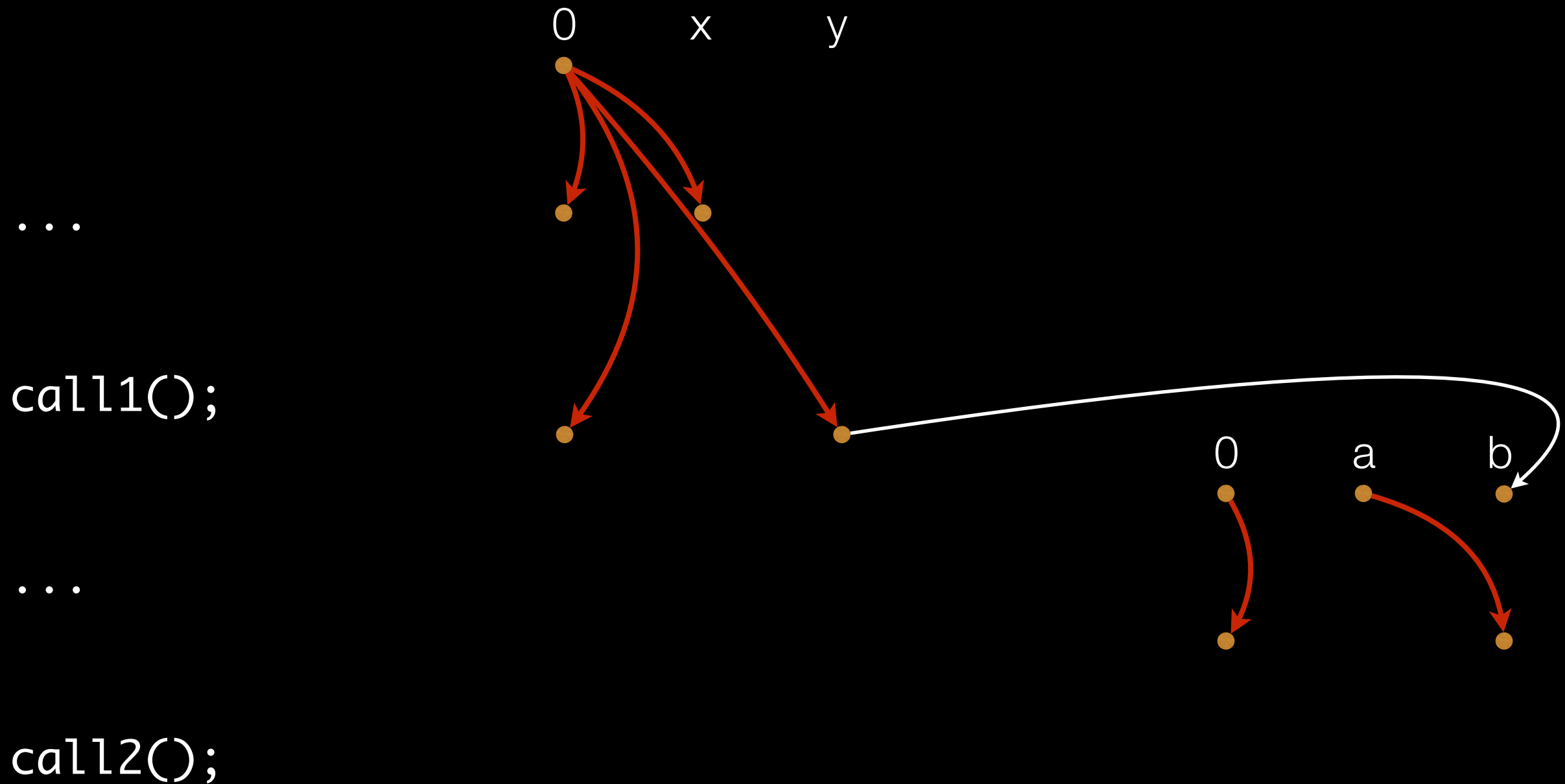- Context Sensitivity?

Method Summaries

Context Sensitivity

0   x   y

...

call1();

...

0   a   b

call2();

Context Sensitivity

0   x   y

...

call1();

...

call2();

0   a   b

Context Sensitivity

0   x   y

...

call1();

...

call2();

0   a   b

Context Sensitivity

0   x   y

...

call1();

...

call2();

Context Sensitivity

0   x   y

...

call1();

...

call2();

0   a   b
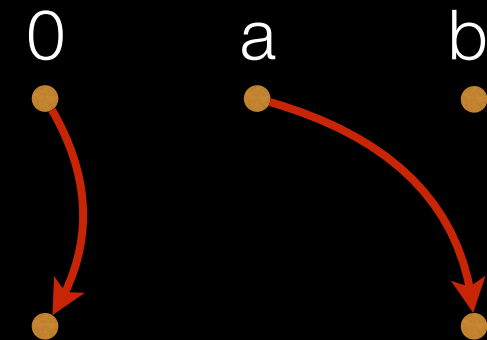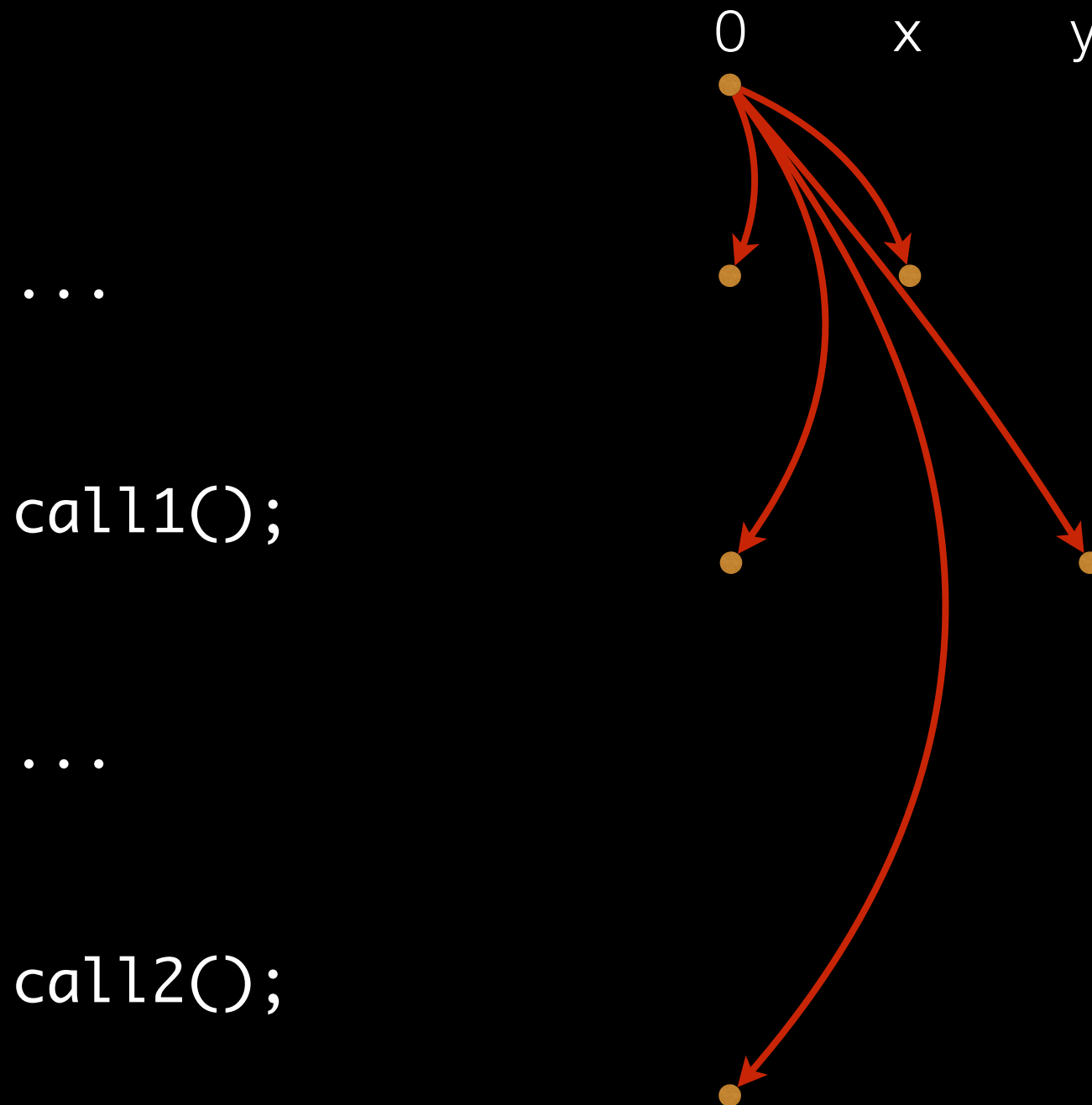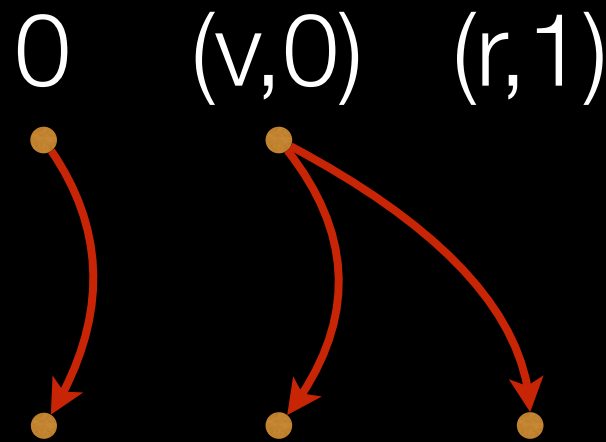
same summary
different contexts => different results

Tidbits

- Flow Sensitivity

- Context Sensitivity

- Large Domains => performance problems

Large Domains

0   (v,0)   (r,1)

r = v + 1
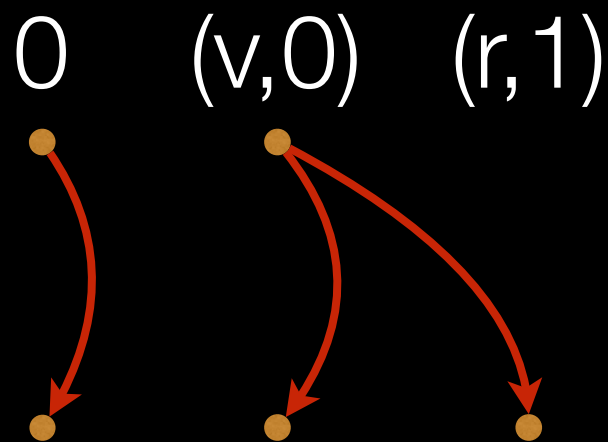
In any context, if *v* is 0 before the call, then it is true that
r is 1 after the call

```
main() {
 x = inc(1);
 y = inc(x);
 z = inc(y);
 print(z);
}
```
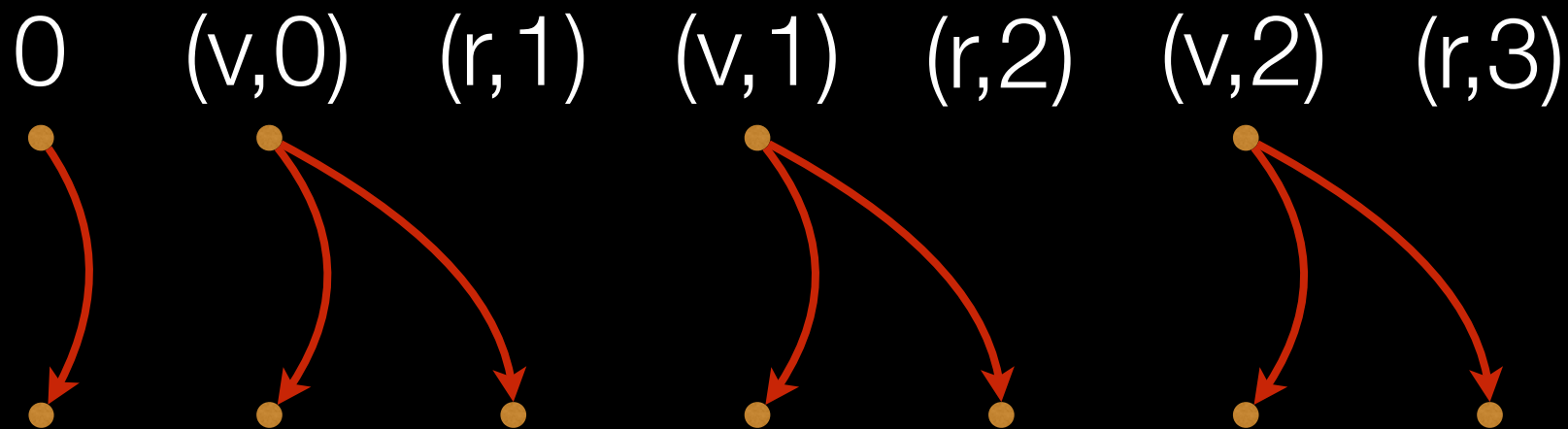
```
inc(v) {
 r = v + 1;
 return r;
}
```

0   (v,0)   (r,1)

Large Domains

```
main() {
  x = inc(1);
  y = inc(x);
  z = inc(y);
  print(z);
}
```
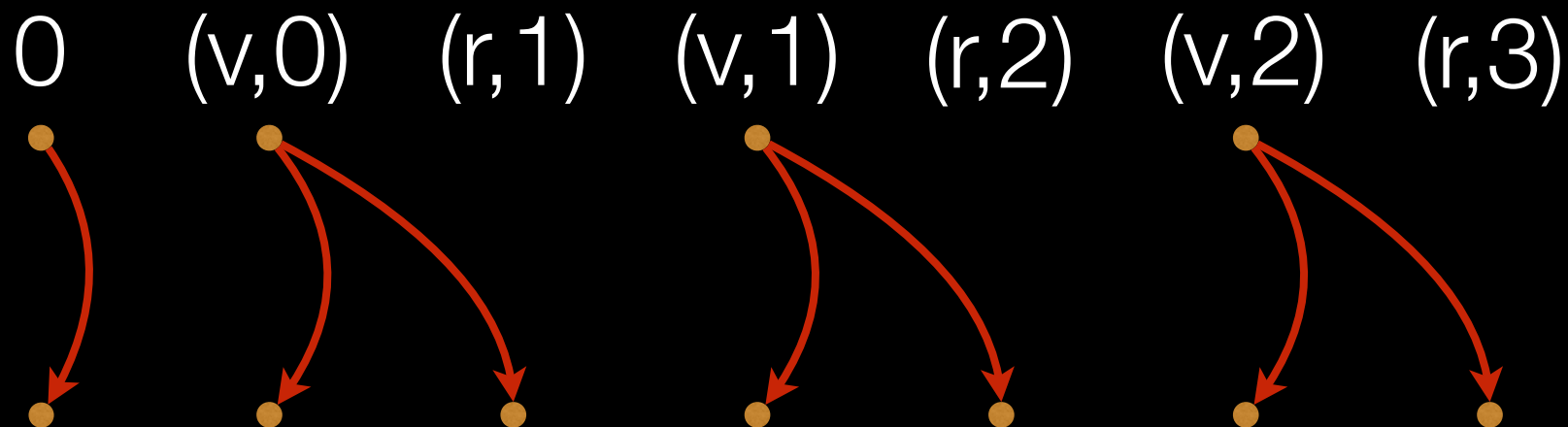
```
inc(v) {
  r = v + 1;
  return r;
}
```

0   (v,0)   (r,1)   (v,1)   (r,2)   (v,2)   (r,3)

```
main() {
  x = inc(1);                    inc(v) {
```

Solution: IDE instead of IFDS

```
}
```

0   (v,0)   (r,1)   (v,1)   (r,2)   (v,2)   (r,3)

Recap

- Interprocedural Finite Distributive Subset

- Flow Functions

- Taint Analysis Example

- On-the-fly ESG

- Tidbits: field-sensitivity, context-sensitivity, large domains

Next

- Context sensitivity